HPI Future SOC Lab – Proceedings 2018

Christoph Meinel, Andreas Polze, Karsten Beins, Rolf Strotmann, Ulrich Seibold, Kurt Rödszus, Jürgen Müller (Eds.)

Technische Berichte Nr. 151

des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam | 151

Christoph Meinel | Andreas Polze | Rolf Strotmann | Ulrich Seibold | Kurt Rödszus | Jürgen Müller (Eds.)

HPI Future SOC Lab

Proceedings 2018

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2023
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

ISBN 978-3-86956-547-7

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
https://doi.org/10.25932/publishup-56371
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-563712

Preface

The *HPI Future SOC Lab* is a cooperation of the Hasso Plattner Institute (HPI) and industry partners. Its mission is to enable and promote exchange and interaction between the research community and the industry partners.

The HPI Future SOC Lab provides researchers with free of charge access to a complete infrastructure of state of the art hard and software. This infrastructure includes components, which might be too expensive for an ordinary research environment, such as servers with up to 64 cores and 2 TB main memory. The offerings address researchers particularly from but not limited to the areas of computer science and business information systems. Main areas of research include cloud computing, parallelization, and In-Memory technologies.

This technical report presents results of research projects executed in 2018. Selected projects have presented their results on April 17th and November 14th 2018 at the Future SOC Lab Day events.

Contents

Service-Oriented Architecture (SOA) and Microservice Architecture (MSA): A Preliminary Survey

Kaushik Rana[1] and Durga Prasad Mohapatra[2]

[1] V.G.E.C, Chandkheda Gujarat, India
Computer Engineering Department
kkr@vgecg.ac.in
[2] N.I.T, Rourkela, Odisha, India
Department of Computer Science and Engineering
durga@nitrkl.ac.in

Web services and microservices are two different concepts of Service-Oriented Architecture, which can be differentiated from its architecture and development pattern. This paper gives more details about web services, microservices and fundamental differences between them.

1 Introduction

In many organizations, it is now acceptable for a business to be architected in much the same way as its technology. Service-Oriented Architecture (SOA) has the ability to architect business, and ultimately changes the way the business is being carried out over the world. This paradigm shift occurs due to closed alignment of business architecture with technology and major advantages offered by SOA over the traditional technology-based business process. SOA enables organizations to align their business process with technology enhancements, changing customer demands creating a flexible and agile business environment.

Microservices is a variant of the service-oriented architecture architectural style that structures an application as a collection of loosely coupled services, which is the fundamental principle of being a service. In microservices architecture, services should be fine-grained and the protocols should be lightweight. This gives the benefit of decomposing an application into different smaller services which improves modularity and makes the application easier to understand, develop and further test. It also parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently. Microservices-based architectures enable continuous delivery and deployment [1].

Basically microservices are designed to cope with failure of large applications which are built on service oriented architecture. Since, multiple unique services are communicating together, it may happen that a particular service fails, but the overall larger applications remain unaffected by the failure of a single service.

The rest of this paper is organized as follows. The next section introduces service-oriented architecture. Section 3 defines web service. Section 4 defines microservice. Section 5 differentiates service oriented architecture and microservice architecture (MSA). And finally, Section 6 concludes the work.

2 What is Service-Oriented Architecture (SOA)?

There are many definitions of SOA but none is universally accepted. We define a few important ones below.

Erl [3] defines service-oriented architecture as a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic. Individually, these units can be distributed.

OASIS [8] defines service-oriented architecture as a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations .

Opengroup [9] defines service-oriented architecture as an architectural style that supports service orientation. Service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services .

Microsoft [6] defines service-oriented architecture as an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology.

Service-oriented architecture is a novel way to design, develop, deploy systems, in which services are reusable business functions exposed via well-defined interfaces, a separation of concern between the interface and its implementation, and service clients use service functionalities from available service descriptions. The SOA infrastructure enables dynamic discovery, dynamic composition, and dynamic invocation of services. From a technical point of view, SOA is an architectural style or design paradigm. Systems that are built based on the SOA principles are called service-oriented systems.

2.1 What is Service?

The central pillar of service-oriented architecture is the *services*. For an SOA, service

- is a reusable component.
- is self-contained, highly modular, autonomous business task.
- is a distributed component.
- has a published service interface.
- stresses interoperability.
- is discoverable.
- is dynamically bound.

2.1.1 Important Service Characteristics

This section describes important characteristics of services [3].

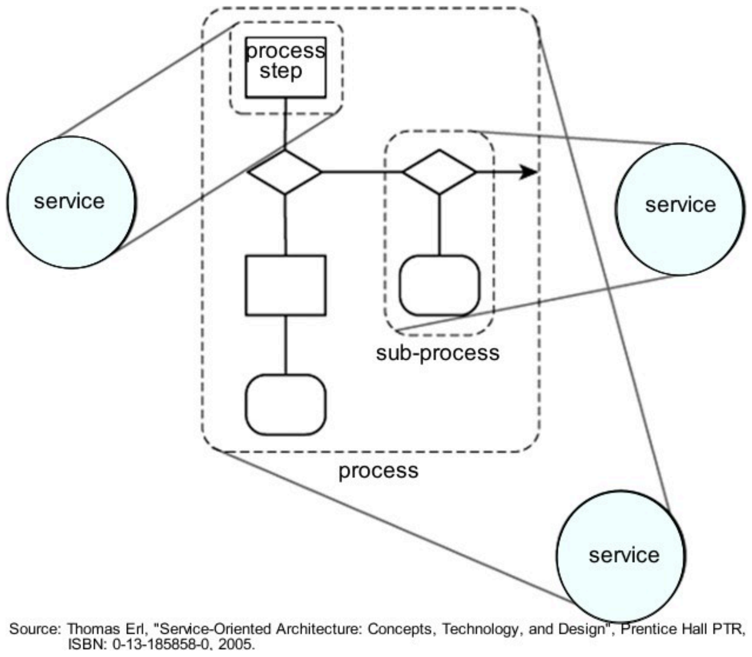

process step
service
service
sub-process
process
service
Source: Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, ISBN: 0-13-185858-0, 2005.

Figure 1: Service encapsulation

Service encapsulations The services encapsulate logic within a distinct context. This context can be specific to a business task, a business entity, or some other logical grouping. The concern addressed by a service can be small or large. Therefore, the size and scope of the logic represented by the service can vary. Further, service logic can encompass the logic provided by other services. In this case, one or more services are composed into a collective.

As shown in figure 1, when building an automation solution consisting of services, each service can encapsulate a task performed by an individual step or a sub-process comprised of a set of steps. A service can even encapsulate the entire process logic. In the latter two cases, the larger scope represented by the services may encompass the logic encapsulated by other services.

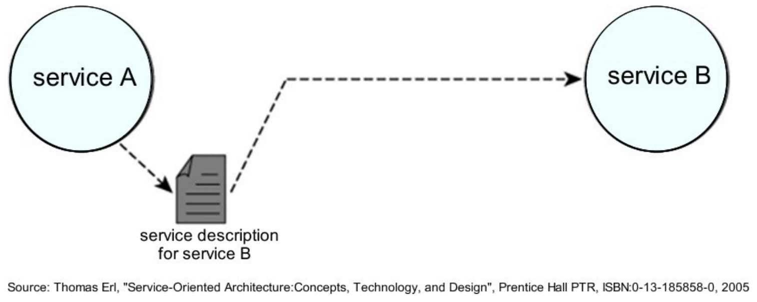

service A
service B
service description for service B
Source: Thomas Erl, "Service-Oriented Architecture:Concepts, Technology, and Design", Prentice Hall PTR, ISBN:0-13-185858-0, 2005

Figure 2: Service relations

Service relations Within SOA, services can be used by other services or other programs. Regardless, the relationship between services is based on an understanding that for services to interact, they must be aware of each other. This awareness is achieved through the use of service descriptions. A service description in its most basic format establishes the name of the service and the data expected and returned by the service. The the manner in which services use service descriptions results in a relationship classified as *loosely coupled*. For example, figure 2 illustrates that service A is aware of service B because service A has acquired B's service description.

Service communications After a service sends a message on its way, it loses control of what happens to the message thereafter. That is why we require messages to be fitted with enough intelligence to self-govern their parts of the processing logic. Hence, messages, like services, should be autonomous and self-governing. Figure 3 shows message existing as an independent unit of communication.

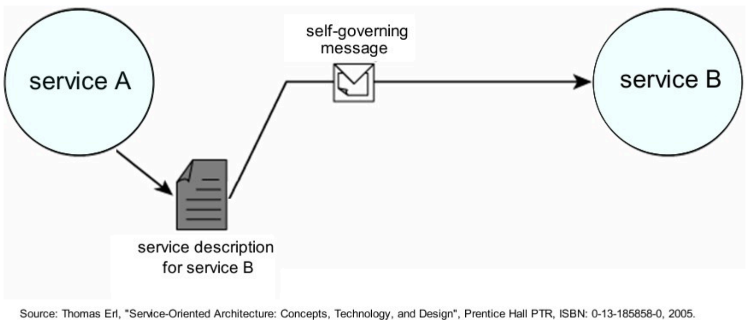

self-governing message
service A
service B
service description for service B
Source: Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, ISBN: 0-13-185858-0, 2005.

Figure 3: Service communications

2.2 Advantages of Service-Oriented Architecture (SOA)

This list of common advantages of SOA is generalized and certainly not complete. It is merely an indication of the potential that SOA can offer [3].

Improved integration (and intrinsic interoperability) SOA can result in the creation of solutions that consist of inherently interoperable services. Because of the vendor-neutral communications framework established by web services-driven SOAs, the potential is there for enterprises to implement highly standardized service descriptions and message structures. The net result is intrinsic interoperability, which turns a cross-application integration project into less of a custom development effort, and more of a modeling exercise.

Inherent reuse Service-orientation promotes the design of services that are inherently reusable. Designing services to support reuse from the get-go opens the door to increased opportunities for leveraging existing automation logic. Building service oriented solutions in such a manner that services fulfill immediate application-level requirements while still supporting the degree of reuse by future potential requesters establishes an environment wherein investments into existing systems can potentially be leveraged and re-leveraged as new solutions are built.

Streamlined architectures and solutions The concept of composition is another fundamental part of SOA. It is not, however, limited to the assembly of service collections into aggregate services. The WS-* platform is based in its entirety on the principle of composability. The reduced performance requirements mentioned previously only refer to the fact that SOA extensions are composable and therefore allow each application-level architecture to contain extensions only relevant to its solution requirements.

Leveraging the legacy investment The industry-wide acceptance of the web services technology set has spawned a large adapter market, enabling many legacies environments to participate in service-oriented integration architectures. This allows IT departments to work toward a state of federation, where previously isolated environments now can interoperate without requiring the development of expensive and sometimes fragile point-to-point integration channels.

Establishing standardized XML data representation On its most fundamental level, SOA is built upon and driven by XML. As a result, an adoption of SOA leads to the opportunity to fully leverage the XML data representation platform. A standardized data representation format (once fully established) can reduce the underlying complexity of all affected application environments. For example, the XML documents and accompanying XML schema definition (XSD) passed between applications or application components fully standardize format and typing of all data communicated. The result is a predictable and therefore easily extensible and adaptable communications network. XML's self-descriptive nature enhances the ability for data to be readily interpreted by architects, analysts, and developers. The result is the potential for data within messages to be more easily maintained, traced, and understood.

Focused investment on communications infrastructure Because web services establish a common communications framework, SOA can centralize inter-application and intra-application communication as part of standard IT infrastructure. This allows organizations to evolve enterprise-wide infrastructure by investing in a single technology set responsible for communication.

"Best-of-breed" alternatives Because SOA establishes a platform-neutral communications framework, it frees IT departments from being chained to a single proprietary development and/or middleware platform. For any given piece of automation that can expose an adequate service interface, you now have a choice as to how you want to build the service that implements it.

Organizational agility Agility is a quality inherent in just about any aspect of the enterprise. Much of service orientation is based on the assumption that what you build today will evolve over time. One of the primary benefits of a well-designed SOA is to protect organizations from the impact of this evolution. When accommodating change becomes the norm in distributed solution design, qualities such as reuse and interoperability become commonplace. The predictability of these qualities within the enterprise leads to a reliable level of organizational agility. However, all of this is attainable through proper design and standardization. Regardless of what parts of service-oriented environments are leveraged, the increased agility with which IT can respond to a business process or technology-related changes is significant.

3 What is Web Service?

Web Service is a remote service communication technology, a way to connect and invoke services altogether into a service-oriented architecture. It is a way to expose the functionality of an application to other application, without a user interface. Moreover it exposes an *Application Programming Interface* (API) over *HyperText Transfer Protocol* (HTTP). The W3C defines a web service as:

> A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards [11].

Web services may use *Simple Object Access Protocol* (SOAP) over HTTP protocol, allowing less costly (more efficient) interactions over the internet. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like SMTP, TCP, UDP, FTP or JMS. In a 2004 document, the W3C extended the definition of web service. There are two major classes of web services: *REST-compliant web services*, in which the primary purpose of the service is to manipulate XML representations of web resources using a uniform set of “stateless” operations; and arbitrary web services, in which the service may expose an arbitrary set of operations [10]. Web services allow applications developed in different technologies to communicate with each other through a common format like XML, Json, etc. Web services are not tied to any one operating system or programming language, i.e, platform neutral. For example, an application developed in Java can communicate with the one developed in C#, Android, etc., and vice versa.

4 What is Microservice?

There is no industry consensus yet regarding the definitions of microservices, and an official definition is missing as well. Some of the definitions that are frequently cited are:

Services in a microservice architecture are often processes that communicate with each other over a network in order to fulfill a goal using technology-agnostic protocols such as HTTP [2]. However, services might also use other kinds of inter-process communication mechanisms such as *shared memory* [2, 4, 12]. Services might also run within the same process. Services in a microservice architecture should be independently developed and deployable. The services are easy to replace [7]. Services can be implemented using different programming languages, databases, hardware and software environments, depending on what fits best. Services are small in size, messaging enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes.

A microservices architecture naturally enforces a modular structure [7]. It lends itself to a continuous delivery of software. A change to a small part of the application only requires one or a small number of services to be rebuilt and redeployed. It adheres to principles such as fine-grained interfaces (to independently deployable services), business-driven development (e.g. domain-driven design), IDEAL cloud application architectures, polyglot programming and persistence, lightweight container deployment, decentralized continuous delivery, and DevOps with holistic service monitoring [2].

4.1 Basic Principles of Microservices

This section describes basic principles of a microservice [5].

Single responsibility This principle implies that a unit, either a class, a function, a module, or a microservice, should have one and only one responsibility. At a point of time, one microservice should have one responsibility, no other.

Built around business capabilities Microservices should focus on specific business function and ensure that it helps in getting things done. A microservice shall never restrict itself from adopting appropriate technology stack or backend database storage which is most suitable for solving the business purpose. This is often the constraint when we design monolithic applications where we try to solve multiple business solutions with some compromises in some areas. Microservices enable you to choose what is best for the problem in hand.

You build it, you own it Another important aspect of such design is related to responsibilities before and after development. In large organizations, usually one team develops the application, and after some knowledge transfer sessions with a maintenance team, it hand over the project to the team. In microservices, the team which build the service owns it and is responsible for maintaining it in future.

This brings developers into contact with the day-to-day operation of their software or microservice and they better understand how their built product is used by customers in real world.

Infrastructure automation Designing and building infrastructure for microservices is another very important need. A service shall be independently deployable and shall bundle all dependencies, including library dependencies, and even execution environments such as web servers and containers or virtual machines that abstract physical resources.

One of the major differences between MSA and SOA is in their level of autonomy. While most SOA implementations provide service-level abstraction, microservices go further and abstract the realization and execution environment.

In traditional application developments, we build a WAR or an EAR, then deploy it into a JEE application server, such as with JBoss, WebLogic, WebSphere, and so on. We may deploy multiple applications into the same JEE container. In an ideal scenario, in the microservices approach, each microservice will be built as a fat Jar, embedding all dependencies and run as a standalone Java process.

Design for failure A microservice shall be designed with failure cases in mind. What if the service fails, or goes down for some time? These are very important questions and must be solved before actual coding starts to clearly estimate how service failures will affect the user experience.

Fail fast is another concept used to build fault-tolerant, resilient systems. This philosophy advocates systems that expect failures versus building systems that never fail. Since services can fail at any time, its important to be able to detect the failures quickly and, if possible, automatically restore service.

Microservice applications put a lot of emphasis on real-time monitoring of the application, checking both architectural elements (how many requests per second is the database getting) and business relevant metrics (such as how many orders per minute are received). Semantic monitoring can provide an early warning system of something going wrong that triggers development teams to follow up and investigate.

4.2 Advantages of Microservice Architecture (MSA)

Microservices offer a number of advantages over the traditional monolithic architectures and SOA. These are [5]:

- With microservices, architects and developers can choose fit-for-purpose architectures and technologies for each microservice. This gives the flexibility to design better-fit solutions in a more cost-effective way.
- As services are fairly simple and smaller in size, enterprises can afford to experiment new processes, algorithms, business logic, and so on. It enables enterprises to do disruptive innovation by offering the ability to experiment and fail fast.

- Microservices enable to implement selective scalability i.e. each service could be independently scaled up or down and the cost of scaling is comparatively less than monolithic approach.
- Microservices are self-contained, independent deployment modules enabling the substitution of one microservice with another similar microservice, when second one is not performing as per our need.
- Because microservices are all about independently manageable services, it enables to add more and more services as the need arises with minimal impact on the existing services.
- With microservices, it is possible to change or upgrade technology for each microservice individually rather than upgrading an entire application.
- As microservices package the service runtime environment along with the service itself, this enables having multiple versions of the service to coexist in the same environment.
- And finally, microservices also enable smaller, focused agile teams for development. Teams will be organized based on the boundaries of microservices.

5 Difference Between Service-Oriented Architecture and Microservice Architecture (MSA): An Example

Let us understand the difference between service-oriented architecture (SOA) and microservice architecture with the help of an example of Online Shopping System (OSS).

In figure 4 the Online Shopping System is developed in service-oriented architecture using web service. In this service, there is only one web service which communicates with database. So this web service might be performing many functional tasks related to database operations.

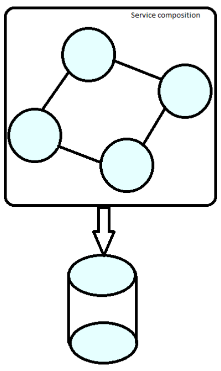


Figure 4: Online Shopping System in SOA

In figure 5 the Online Shopping System is developed in microservice architecture using microservices. All the components of the services are developed independently, single functional responsible, fine-grained clearly scoped microservices. In general, in SOA a service may be composed of other services; in MSA a service is independent and self-contained, which means it cannot be composed of other services. Table 1 shows small comparison between SOA and MSA.

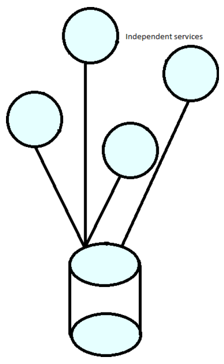


Figure 5: Online Shopping System in MSA

6 Conclusions

This paper gives more details about web services, microservices and fundamental difference between them. We implemented prototype web services over the 1000 node cluster accessible through the HPI Future SOC Lab at the *Hasso Plattner Institute*, Germany; still more work is required to develop large, complex both web services and microservices. In the future, we will implement large sized service, run and will measure their performance over large scale cluster.

Table 1: Comparison between SOA and MSA

Sl. No.	SOA	MSA
1.	No independence of services	Independent services
2.	Multiple responsibilities within a service	Single responsibility per service
3.	Services are composed	Service are not composed.

References

[1] L. Chen and M. A. Babar. “Towards an Evidence-Based Understanding of Emergence of Architecture through Continuous Refactoring in Agile Software Development”. In: *Proceedings of the 11th IEEE/IFIP Conference on Software Architecture*. IEEE Computer Society, 2014, pages 195–204. ISBN: 978-1-4799-3412-6. DOI: 10.1109/WICSA.2014.45.

[2] N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. “Microservices: yesterday, today, and tomorrow”. In: *CoRR* abs/1606.04036 (2016). arXiv: 1606.04036. URL: http://arxiv.org/abs/1606.04036.

[3] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005. ISBN: 978-0-13-185858-9.

[4] M. Fowler. *Microservices*. 2017. URL: https://martinfowler.com/articles/microservices.html.

[5] L. Gupta. *Microservices – Definition, Principles and Benefits*. 2017. URL: https://howtodoinjava.com/microservices/microservices-definition-principles-benefits/.

[6] Microsoft. *Microsoft service oriented architecture (soa)*. 2017. URL: https://msdn.microsoft.com/enus/library/bb833022.aspx..

[7] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen. *Microservice Architecture: Aligning Principles, Practices, and Culture*. 1st. O’Reilly Media, Inc., 2016. ISBN: 1-4919-5625-9.

[8] OASIS. *Organization for the advancement of structured information standards*. 2017. URL: https://www.oasisopen.org..

[9] Opengroup. *Service oriented architecture : What is soa?* 2017. URL: http://www.opengroup.org/soa/sourcebook/soa/soa.htm..

[10] W3C. *Web services architecture & relationship to the world wide web and rest architectures*. URL: https://www.w3.org/TR/ws-arch/.

[11] W3C. *Web services glossary & web service*. URL: https://www.w3.org/TR/ws-gloss/.

[12] E. Wolff. *Microservices: Flexible Software Architecture*. Addison-Wesley, 2017. ISBN: 978-0-13-465044-9.

Tetris of Big Spatial Data

Julia Sidorova[1], Lars Lundberg[1], and Lars Sköld[2]

[1] Blekinge Institute of Technology
lars.lundberg@bth.se
[2] Telenor Swerige AB
Lars.Skold@telenor.se

The population in Sweden is growing rapidly due to immigration. In this light, a dilemma emerges in telecommunications: either new antennas can be installed at hot spots of user demand, which will require an investment, and/or the clientele expansion can be carried out in a planned manner to promote the exploitation of the infrastructure in the less loaded geographical zones. In the previous work [4, 5, 6, 7], we proposed Tetris of spatial data, which is a linear programming formulation for the footprints that different user segments produce on the network restricted by antennas' capacities, and solved the dual (the first option in the dilemma) and the primal (the second one). The work is done on real data in an academia—industry collaboration between Blekinge Institute of Technology and Telenor, a major Scandinavian operator. The past period has been the finalizing stage of this project, during which we have addressed two problems: 1) will the conclusions change with time resolution?, and 2) is there a way to get around the prohibitive complexity? Most of the code and the foot- print matrix are available on SourceForge. We also discuss the implications of the GDPR for a project involving user's spatial data.

1 Project Idea

Apart from the Tetris formulation (dual and primal), in the previous work we have addressed:

1. the ways to segment the clients into geo-demographic groups based on their mobility history: via trajectory clustering [5] and based on their relative mobility concerning the location of the rest of the population [7];
2. the creation of human-readable queries and data summary based on fuzzy logic [4];
3. alternatives of the optimal implementation with noSQL databases [3].

During the past period we have experimentally addressed the question: shall the conclusions about the superiority of the Tetris approach over the default strategy still hold, if we change the time resolution?

Via a simulation we have gone from the original 5 min resolution to the 10 min one, and the global conclusion has not changed that Tetris is recommendable in place of the default non-optimized business expansion [8]. Also this time a feasibility of the implementation in big data contexts has been under scrutiny, since Tetris is formulated as an integer linear programming problem, which is generally NP-hard. We have shown that a classical remedy of relaxing the integer restrictions works.

The rest of the paper is organised as follows. Section 2 describes the data set. In Section 3, the linear programming formulation at the heart of Tetris is explained. In Section 4, the experiments are reported to explore whether the conclusions were dependent on the time resolution. Section 5 outlines our approach in handling the complexity obstacle. The conclusions of the project are drawn in Section 6. Finally, in Section 7 we give our view on the implications of the new EU legislation to a project like Tetris.

2 Data

The study has been conducted on anonymized geospatial and geo-demographic data provided by a Scandinavian telecommunication operator. The data consist of CDRs (Call Detail Records) containing historical location data and calls made during one week in a mid-size region in Sweden with more than one thousand radio cells. Several cells can be located on the same antenna. The cell density varies in different areas and is higher in city centers, compared to rural areas. The locations of 27 010 clients are registered together with which cell serves the client. The location is registered every five minutes. During the periods when the client does not generate any traffic, she does not make any impact on the infrastructure and such periods of inactivity are not included in the resource allocation analysis. Every client in the database is labeled with her geo-demographic segment. The fields of the database used in this study are:

- the cells IDs with the information about which users it served at different time points,
- the location coordinates of the cells,
- the time stamps of every event (5 minute resolution),
- the MOSAIC geo-demographic segment for each client, and
- the Telenor geo-demographic segment for each client.

There are 14 MOSAIC segments present in the database. The six in-house Telenor segments were developed by Telenor in collaboration with InsightOne, and, to our best knowledge, though not conceptually different from MOSAIC, they are especially crafted for telecommunication businesses.

3 Tetris of Spatial Data

As input the combinatorial optimization module takes the matrix $\mathbf{A}$ of size $N \times M$, where N is the number cell towers × number of time slots (there are 2016 five minute slots in one week), and M number of geo-demographic segments. Every cell $\mathbf{A}_{i,j}$ contains the footprint by the users from geo-demographic segment j. As output the recommendation is given about the optimal proportion of geo-demographic segments in the customer base. An LP formulation (the primal) is given to the problem is as follows:

- *the objective function* maximises the number of clients,
- *the decision variable*s are the scaling coefficients to boost or to reduce the number of clients in each of the segments, and
- *the restrictions* are formulated not to overload any cell at any time, i.e. that the historical footprint of the segment scaled with the coefficent is less than the cell's capacity.

Let us define the model's variables:

- Ik: *the set with geo-demographic segments* $\{segment1, \ldots, segmentk\}$;
- D: the mobility data for a region that for each user contain a client's ID, a client's geo-demographic segment, time stamps when the client generated traffic, and which antenna served the client at a particular time stamp.
- S_i: the number of subscribers that belong to a geo-demographic segment i;
- $S_{i,t,j}$: the number of subscribers that belong to a geo-demographic segment i, who are using the network at some time moment t, being registered with a particular cell j;
- C_j: the capacity of cell j in terms of how many persons it can safely handle simultaneously;
- $\mathbf{x}$: the vector with the scaling coefficients for the geo-demographic segments.
- *The vector* $\mathbf{x}$ *with the decision variables* $\mathbf{x} = \{x_{segment1}, \cdots, x_{segmentM}\}$.
  The decision variables represent the scaling coefficients for each geo-demographic segment. For example, for the category in the clientele that is to be doubled $x_i = 2$. Similarly, if $x_i < 1$ for a geo-demographic segment, it means that the number of clients is to be reduced.
- *The objective function* seeks to maximize the number of subscribers:
  $$Maximize \qquad \textstyle\sum_{i=1\ldots15} S_i x_i.$$
- *The restrictions*
  $$for\ all\ \ t, j, \textstyle\sum_{i=1\ldots15} S_{i,t,j} x_i \leq C_j$$

Since the mobility pattern for the subscribers is predictable due to strong spatio-temporal regularity, the increase in the number of subscribers in a given segment with a factor x will result in an increase of the load generated by the segment with a factor x for each time and cell.

The value of the objective function is sensitive to the number of segments: the more segments, the more degrees of freedom the LP has, and the higher the objective value is. There are 13 geo-demographic MOSAIC segments present in the database. A good decision support discovers and prompts actions that lead to revenues, and different actions suggested are comparable profit-wise: among competing modules, the one that reveals more profitable actions is better. Even when the optimal proportion of segments in the customer base can be unreachable for various reasons, the optimal proportions indicate a direction of the maximum revenue and most efficient infrastructure exploitation.

The dual of the described above primal is the problem of optimal cell expansion [8].

4 Are the conclusions dependent on time resolution?

In the previous work we have experimentally confirmed the common sense hypothesis that more degrees of freedom (more segments) lead to a more beneficial solution, but how about the time resolution? The answer is that in principle the conclusions do not change, as is depicted in Figure 1 and in greater detail described in [8].

5 Relaxing integer restrictions

Generally, integer linear programming problems are NP-hard [9], thus making them infeasible in big data contexts. The standard way to avoid the infeasibility problem is to relax the integer linear programming problem by removing the integer restriction. In [8] we show that such a relaxation is possible and we do not loose any important information, which means that the proposed solution is applicable in big data scenarios.

6 Conclusions

The conclusions of the Tetris of big spatial data are as follows.

1. The Tetris of big spatial data in the primal form suggests an optimal proportion (a healthy equilibrium) of geo-demographic segments in the customer base, given the infrastructure is exploited in the optimal way. The dual form solves the problem of intelligent infrastructure growth.

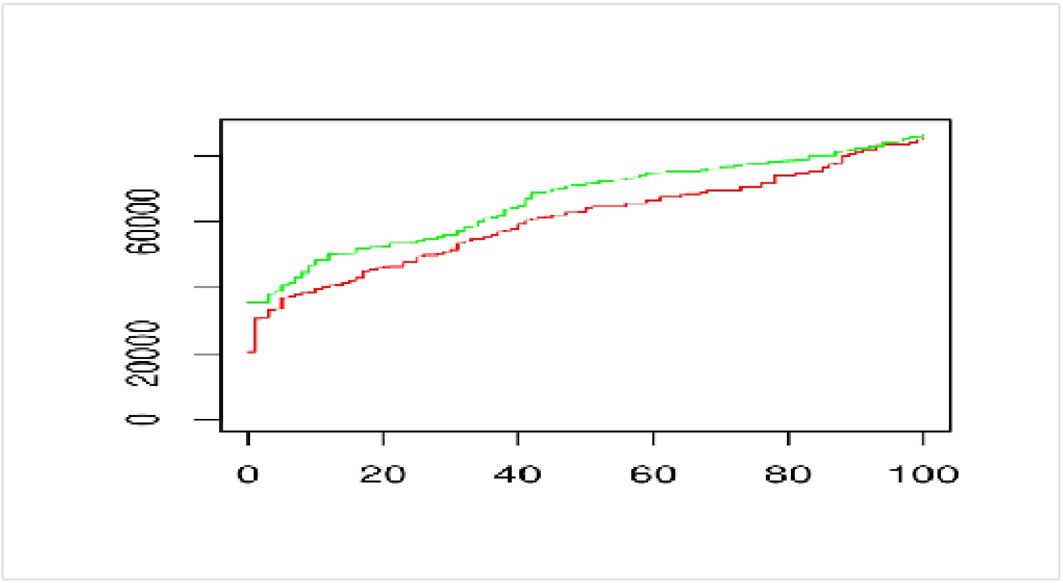


Figure 1: The optimized (green) and the default (red) customer base expansion. The X axis is antenna capacity and Y is the number of allocated clients.

2. There are analytical questions in which user-individual segmentations based on trajectory clustering is a preferable modeling compared to geo-demographic segmentations. The latter is still a handy user "handle" with clear advantages and essentially the Tetris was formulated for MOSAIC/Telenor geo-demographic segments. The approach based on relative mobility led to the definition of infrastructure-stressing clients and complements the solution based on geo-demographic segments [6]. The infrastructure-stressing client is the one, who predominantly stays in the zones of the network's high load, "a hot spotter", in other words. This property is inherently linked to the mobility of other users and the characteristics of the infrastructure in a particular region. A list of such clients has been revealed in the customer base. Being infrastructure-stressing has turned out to be independent of the person's geo-demographic segment.

3. The mission of fuzzy logic in the era of big data is to summarize multitudes of data into a relevant human-readable conclusion, so we have designed queries like "Tell me how successful the operator has been in a specific region: highly/rather/not at all".

7 The implications of the new GDPR

It would be natural in the era of personalized medicine and personalized web searches to think of personalized tariffs and to have slightly different tariffs for infrastructure-friendly (cheaper) and infrastructure-stressing clients (more expensive). Yet, during the project's life circle, EU legislation has changed considerably with the General

Data Protection Regulation coming into force in May, 2018. In our understanding, Article 89 (1) and (2) "Scientific and historical research purposes or statistical purposes" are of relevance to a research project like Tetris. "Recital 159 states that scientific research should be 'interpreted in a broad manner' and includes privately funded research as well as studies carried out in the public interest. In order for processing to be considered statistical in nature, Recital 162 says that the result of processing should not be 'personal data, but aggregate data' and should not be used to support measures or decisions regarding a particular individual", say the legal text as cited in [1]. Therefore, the application of personalized tariffs is not legal, since the result of the method is a decision regarding an individual user, whether to her benefit or detriment. Nonetheless, the infrastructure-stressing client does not necessarily start to be an illegal concept, because it can be used as an aggregate value for the purposes of internal analytics, for example, to see whether the percent of the IS clients has gone down in the assessment of the success of the infrastructure updates. Other analytics based on mobility data are less extreme than the infrastructure-stressing clients and, can be analogously reformulated to become aggregate values, as well.

There is a more general question: legally speaking, are mobility data "personal data"? In our understanding, the answer is positive, because of the following argument. According to [1], personal data is defined as "any information relating to an identified/identifiable natural person, a 'data subject'. A data subject is a natural person, who can be identified, or is identifiable, directly or indirectly". In the literature, it was demonstrated that 87 % of the US population are uniquely identifiable by postal code, date of birth and gender, and it is easy to see how it can be inferred from a stream of location data [9]. According to [2], four spatiotemporal points are enough to uniquely identify 95 % of individuals, and it was shown that making the datasets significantly coarser does not bring a decisive effect on reducing reidentifiability. A wider discussion about the challenges in the management and use of location data including privacy challenges such as re-identification and inference can be found in [10]. A lot of knowledge beneficial to the society seems to be obtainable from location data in a fairly legal way, for example, as a next step we will do the optimal location of ambulances in the region and the location of bus stops.

Acknowledgements

This work is part of the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden. For the past two years, the experiments were executed on the machines at Hasso Plattner Institute.

References

[1] Bird & Bird, editor. *Guide to the General Data Protection Regulation*. May 2017. URL: https://www.twobirds.com/-/media/pdfs/gdpr-pdfs/bird--bird--guide-to-the-general-data-protection-regulation.pdf (last accessed 2018-03-17).

[2] Y. De Montyoye, C. A. Hidalgo, M. Verleysen, and V. D. Belondell. "Unique in the crowd: the privacy bounds of human mobility". In: *Scientific reports* 3 (2013), page 1376.

[3] C. Niyizamwiyitira, L. Sköld, L. Lundberg, and J. Sidorova. "Analytic queries on Telenor data". In: *HPI Future SOC Lab: Proceedings 2016*. Edited by C. Meinel, A. Polze, G. Oswald, R. Strotmann, U. Seibold, and B. Schulzki. Potsdam, Germany, 2016.

[4] S. Podapati, L. Lundberg, L. Sköld, O. Rosander, and J. Sidorova. "Fuzzy Recommendations in Marketing Campaigns". In: *IISA 2017 The 8th IEEEh International Conference on Information, Intelligence, Systems and Applications. 28-30 August, Larnaca, Cyprus*. 2017. DOI: 10.1007/978-3-319-67162-8_24.

[5] S. Sagar, L. Sköld, L. L., and S. J. "Trajectory Segmentation for a Recommendation Module of a Customer Relationship Management System". In: *The 2017 Int. Symposium on Advances in Smart Big Data Processing*. 2017. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.177. Forthcoming.

[6] J. Sidorova, O. Rosander, L. Sköld, H. Grahn, and L. Lundberg. "Finding a Healthy Equilibrium of Geo-demographic Segments for a Telecom Business: Who Are Malicious Hot-Spotters?" In: *Machine Learning Paradigms*. Springer International Publishing, July 2018, pages 187–196. DOI: 10.1007/978-3-319-94030-4_8.

[7] J. Sidorova, L. Sköld, O. Rosander, and L. Lundberg. "Recommendations for Marketing Campaigns in Telecommunication Business based on the footprint analysis". In: *The 1st International Workshop on Data Science: Methodologies and Use-Cases (DaS 2017) at 21st European Conference on Advances in Databases and Information Systems (ADBIS 2017), 28-30 August, Larnaca, Cyprus*. LNCS. DOI: 10.1109/IISA.2017.8316396.

[8] J. Sidorova, L. Sköld, O. Rosander, and L. Lundberg. "Optimizing utilization in cellular radio networks using mobility data". In: *Optimization and Engineering* 20.1 (May 2018), pages 37–64. DOI: 10.1007/s11081-018-9387-4.

[9] L. Sweeney. *Simple demographics often identify people uniquely*. Carnegie Mellon University, Data Privacy Working Paper 3. 2000.

[10] The British Academy and The Royal Society, editors. 2017. Chapter 6: Personal Location Data. URL: https://www.thebritishacademy.ac.uk/documents/1886/Data-management-use-case-studies-technologies-governance.pdf.

CDI: A Complex Event Processing Distributed System for Time Series Forecasting Applied to Air Quality

Luís Fernando L. Grim[1,2] and André Leon S. Gradvohl[2]

[1] Federal Institute of Education, Science and Technology of São Paulo (IFSP), Piracicaba–SP, Brazil
lgrim@ifsp.edu.br

[2] University of Campinas (FT/UNICAMP), Limeira–SP, Brazil
School of Technology
gradvohl@ft.unicamp.br

The Online Sequential Extreme Learning Machine algorithm is suitable for forecasting data streams with Concept Drifts. Nevertheless, data streams forecasting require highperformance implementations due to the high incoming samples rate. In this preliminary work, we analyzed the scalability of parallel implementations for the Online Sequential Extreme Learning Machine in C programming language, with OpenBLAS, Intel MKL, and MAGMA libraries that we proposed. Thus, the goal of this work is to compare the performance — prediction error and execution time — of the proposed implementations when forecasting concentrations of breathable Particulate Matter in the air, on the resources that the HPI Future Soc Lab made available to us. Based on the results obtained in this work, the future intention is to construct an ensemble that distributes several models with one of the implementations developed as kernels, functioning as operators of a Complex Event Processing System. Experimental results showed that, in the cases approached here, the update block size of the algorithm and the execution environment influences the real execution time of each implementation differently.

1 Introduction

Forecasting the concentration of air pollutants has drawn the attention of public authorities and environmental agencies due to the occurrence of several pollution episodes in urban areas around the world. High concentrations of pollutants cause harmful health effects and even premature deaths among sensitive groups, such as children, the elderly and people with respiratory and cardiac diseases [3].

We consider the datasets of air pollutants' concentrations as Time Series because they correspond to a collection of values obtained periodically. In addition, we regard these datasets as data streams, since they are an ordered sequence of instances arriving at high speed, i.e. a high number of instances per unit of time. Considering these characteristics, the systems that deal with data streams cannot handle many instances in main memory for an extended period, or they may experience a memory overload. Therefore, we need real-time processing to deal with data streams.

Data streams are dynamic by nature and may present Concept Drifts, that consists in changing the relationship between the input data and the target variable that someone would like to predict over time [4]. This dynamic nature may compromise the model performance when predicting new incoming samples. Online learning approaches, like Online Sequential Extreme Learning Machine (OS-ELM) [10], can deal with concept drift scenarios, due to the capability to learn new data patterns over time, making it suitable for data streams processing.

In our work, we propose parallel implementations of OS-ELM developed in C, to run in multicore CPUs and hybrid architectures, i.e. CPU + Graphical Processing Units (GPU). Therefore, our goal is to compare the performance of these implementations when forecasting breathable Particulate Matter (PM_{10}) concentrations in the air for future use on an ensemble that distributes several models, functioning as operators of a Complex Events Processing (CEP) System.

We organized the remainder of this paper as follows. Section 2 describes works related to the improvement of OS-ELM performance and PM forecasting. Section 3 describes our implementations in detail and presents the methodology adopted, including the dataset description, preprocessing steps, experimental setup and the resources used. Finally, section 4 discusses the experimental results. Section 5 concludes this work and discusses possible directions for future works.

2 Related Works

Currently, several researchers have been dedicated to the study of predicting PM_{10} and other air pollutants. In [12], the authors developed an ensemble of Artificial Neural Networks (ANNs) to predict daily concentrations of PM_{10}, which showed better performance when compared to single ANNs. In a recent work, [14] compared an Extreme Learning Machines (ELM) [7] with a Support Vector Machine (SVM) algorithm for the classification task on PM concentration dataset. Results showed that ELM outperformed the SVM approach in accuracy and processing time.

In [2], the authors compared online sequential learning approaches based on OS-ELM with offline approaches based on the original ELM when predicting PM concentrations. Results showed not only that online sequential learning offers better accuracy in such scenarios, but also that ensembles of OS-ELMs improve the stability of the results. However, in general, the computational resources required by the ensembles increased as well.

In [1] the authors presented a comprehensive approach to the use of a tool called High-Performance Extreme Learning Machines (HP-ELM). HP-ELM has a version that provides acceleration through the Matrix Algebra library on GPU and Multicore Architectures (MAGMA) [13]. The accelerated parts are the calculation of the correlated matrices and the vector of output weights. The results showed that acceleration in the GPU benefits problems that use more than 1,000 hidden neurons.

Following the same reasoning, [9] proposed an approach that accelerates the OS-ELM algorithm through GPUs. The author did the experiments in R environment with RMOA, using the cuBLAS library [11] and the following input parameters: from

10 to 100 hidden neurons with sigmoid activation function and a 2,500 instances block size. The results showed that the simple delegation of the matrix operations to the GPU reduced the processing time almost 10-fold.

Based on the previous works, the OS-ELM seems suitable to forecast PM concentrations in data streams scenarios. Nevertheless, this approach spends more time when compared with offline models like ANNs and ELMs, especially when using ensembles. Acceleration in GPUs of the HP-ELM and OS-ELM with MAGMA and cuBLAS libraries showed promising results for models with the respective parameters presented. In our work, we decided to explore implementations of the OS-ELM in C programming language using the most consolidated libraries for matrix operations (OpenBLAS [15], Intel MKL [8], and MAGMA). Using different input parameters and execution environments, we intend to analyze the performance of all developed implementations, as well as the scalability factor of some of them.

3 Proposed Implementations

Using the OS-ELM in MATLAB [6] as a base, we adapted our proposed implementations to data stream processing considering a dataset where we chose a chunk of size N_0 for the initial training, and the remaining instances simulated an online stream, which were predicted by the forecasting model.

We have developed three distinct C implementations using the following libraries: OS-ELM with OpenBLAS (OS-ELMob), OS-ELM with MKL (OS-ELMmkl) and OS-ELM with MAGMA (OS-ELMmgm). These libraries enable us to perform the matrix operations using Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK) functions. OpenBLAS and Intel MKL provide functions that explore the multithread features in multicore processors, which also enables the parallelism in multiprocessors architectures. In turn, MAGMA offers functions that run parallel in heterogeneous/hybrid architectures for Multicore + GPU systems. In this work, we restricted the proposed implementations in C programming language to the cases of regression with the sigmoid activation function ($g(x) = \frac{1}{(1+e^{-x})}$).

3.1 The dataset and pre-processing steps

The environmental dataset used in the experiments contains samples of PM_{10} concentration collected sequentially over time. These concentrations are measured in micrometers per cubic meter ($\mu m/m^3$), and the QUALAR system provided them. The dataset contains hourly collected samples from 01 January 1998 to 23 November 2017 from the Cubatão—Vila Parisi automatic station. The dataset comprises 174 397 samples, presents zero as the minimum value, 1470 $\mu m/m^3$ as the maximum value, and 8011 missing samples. To mitigate the effects of these missing samples, we filled them with the most probable value, using "Amelia II" [5] and linear interpolation.

We did an outlier analysis considering maximum values of PM_{10} reported by the CETESB for the Cubatão—Vila Parisi monitoring station. We normalized about 964 samples above 350 $\mu m/m^3$. Then, we used the Hampel filter for the detection and replacement of the remaining outliers in the dataset. Finally, we normalize the data between $[0,1]$, as suggested by [6], by the min-max normalization.

In the end, we organized hourly samples in such a way that the current sample and the last five instants of time, $x_n = [s_{n-5} \dots, s_{n-1}, s_n]$, are used to predict the next instant $y = [s_{n+1}]$. In other words, the current and the last five hourly measured concentrations are used to forecast the concentration of the next hour.

3.2 Configurations of the OS-ELM Implementations

We tested cases with 100 hidden neurons and with update blocks of 1, 10, 100, 1000, 2500 and 5000 instances. For the initial training dataset, we used the 5000 older instances, and we simulated a data stream with the remaining 169 397 instances. For the cases tested, we evaluated the performance of each implementation measuring the prediction error and the processing time through the means and the standard deviations. As a measure of prediction error, we considered the Root Mean Square Error (RMSE) between the actual and predicted outputs. Regarding the time, we consider the real times (in seconds) to processing the entire algorithm. We repeated the experiments for 50 trials,

3.3 Used Future Soc Lab Resources

We conducted all tests in two distinct HPI Future SOC Lab environments. The first of them is a GPU Cluster with 32 cores Intel (R) Xeon (R) CPU E5-2620 v4 @ 2.10 GHz; 128 GB of RAM; 6 Tesla K80 @ 824 MHz and 11,440 MBytes; and Ubuntu 16.04.2 LTS 64 bits O.S. In the GPU cluster, we installed the OpenBLAS 0.2.20 and Intel MKL 2018.1.163 libraries; both compiled with the default configuration, which limited the use of our implementations with these libraries up to 16 threads. We also installed the MAGMA 2.2.0 library, compiled for CUDA 8.0 and OpenBLAS 0.2.20. One should notice that, although cluster has 6 GPUs, for now our OS-ELMmgm implementation is restricted to the use of only one GPU.

The second environment in which we ran the tests consists of a set of Virtual Machines (VMs) with virtual CPUs (vCPUs) Intel Core i7 9xx (Nehalem Class Core i7) @ 2.39 GHz; and Ubuntu 16.04.2 LTS 64 bits O.S. The differences between them are the number of vCPUs, the amount of RAM and the amount of storage space, as follows:

- Small: 1 vCPU, 1 GB RAM, 5 GB HDD;
- Medium: 2 vCPUs, 2 GB RAM, 20 GB HDD;
- Large: 4 vCPUs, 4 GB RAM, 40 GB HDD;
- Extra Large (XL): 8 vCPUs, 8 GB RAM, 80 GB HDD;

In all VMs, we also installed the OpenBLAS 0.2.20 and Intel MKL 2018.1.163 libraries, compiled with the default configuration. For the VMs, the default number of threads given by the libraries always follows the number of cores of the respective VM.

4 Experimental Results

Initially, we tested the OS-ELMob, OS-ELMmkl, and OS-ELMmgm implementations on the GPU cluster and the OS-ELMob and OS-ELMmkl implementations on the VMs. However, for some unknown reason, OS-ELMob implementations presented substantial predictive errors, and in many cases even internal routines errors for Medium, Large and XL VMs. Therefore, we consider only the results of the OS-ELMmkl implementations for the VMs.

Table 1 shows the average RMSEs and their respective standard deviations (SD) when varying the Block Size (BS) of each implementation for the execution in GPU Cluster. Table 2 shows the same of OS-ELMmkl implementation for the execution in each VM.

The scenario presented in table 1 and table 2 indicates that the RMSEs are practically the same in all implementations (0.0001), regardless of the execution environment, as well as the BS and also related to the VM size employed.

Table 3 presents the mean and SD of the Total Real Time (TRT) required for processing the whole data stream, comparing OS-ELMob and OS-ELMmkl, that are multithread implementations and OS-ELMmgm, that runs in a hybrid architecture, all executing on GPU cluster.

Notice that, we did not make additional configurations related to the number of threads, so all implementations ran with their respective default values, which was 16 threads for the configured environment on CPU, one for each core.

Table 3 shows that OS-ELMob presented better TRTs for cases with BSs up to 100 instances, where OS-ELMob ran about 13.5-fold faster than OS-ELMmgm for the case with BS=1. However, OS-ELMmgm shows to be superior for cases with BSs above 1,000 instances, running about 2.4-fold faster than OS-ELMmkl and 3.2-fold faster than OS-ELMob when BS=5,000.

To better visualize the speedups compared to the OS-ELMmkl for the execution in GPU cluster, figure 1 presents the speedups in TRT. In addition, figure 1 shows that the OS-ELMob and OS-ELMmkl perform better for smaller BSs, which have more loop iterations and less complex matrix operations. With OS-ELMmgm the opposite happens: it performs better with larger BSs, which have fewer loop iterations and more complex matrix operations. However, even for the largest BS analyzed here, which has 5,000 instances, the OSELMmgm speedup just about 2.5-fold when compared with OS-ELMmkl.

Table 4 presents the mean and SD of the Total Real Time (TRT) required for processing the whole data stream for OS-ELMmkl in VMs. Figure 2 presents the speedups concerning the Small VM for the OS-ELMmkl executed in others VMs. It is important to mention that in all cases covered by table 4 and figure 2, the OS-ELMmkl runs with one thread per core, according to VM. Then in Small VM OS-ELMmkl runs

Table 1: RMSEs for each case executed on GPU cluster

BS	OS-ELMob		OS-ELMmkl		OS-ELMmgm	
1	0.108 00	± 0.000 03	0.108 00	± 0.000 04	0.108 00	± 0.000 04
10	0.108 00	± 0.000 03	0.108 00	± 0.000 04	0.108 00	± 0.000 05
100	0.108 00	± 0.000 04	0.108 00	± 0.000 04	0.108 00	± 0.000 05
1000	0.108 10	± 0.000 05	0.108 10	± 0.000 05	0.108 10	± 0.000 05
2500	0.108 10	± 0.000 03	0.108 10	± 0.000 03	0.108 10	± 0.000 04
5000	0.108 10	± 0.000 05	0.108 10	± 0.000 04	0.108 10	± 0.000 05

Table 2: RMSEs for each case executed on VMs

BS	Small	Medium	Large	XL
1	0.1080	0.1080	0.1080	0.1080
10	0.1080	0.1080	0.1080	0.1080
100	0.1080	0.1080	0.1080	0.1080
1000	0.1081	0.1081	0.1081	0.1081
2500	0.1081	0.1081	0.1081	0.1081
5000	0.1081	0.1081	0.1081	0.1081

[1] As in table 1, the SD of all cases varied between 0.00003 and 0.00005.

Table 3: TRTs for each case executed on GPU cluster

BS	OS-ELMob	OS-ELMmkl	OS-ELMmgm
1	9.90 ± 0.47	13.42 ± 0.88	134.95 ± 0.70
10	2.48 ± 0.02	9.47 ± 1.20	17.13 ± 0.07
100	2.29 ± 0.02	6.98 ± 0.90	4.27 ± 0.02
1000	10.31 ± 0.07	11.77 ± 0.39	5.09 ± 0.03
2500	34.92 ± 0.15	34.17 ± 0.62	12.08 ± 0.08
5000	105.33 ± 1.43	78.04 ± 1.69	33.08 ± 0.17

Table 4: TRTs for each case of OS-ELMmkl executed on VMs

BS	Small	Medium	Large	XL
1	5.07 ± 0.07	5.32 ± 0.59	5.20 ± 0.35	6.81 ± 0.24
100	3.54 ± 0.09	2.88 ± 0.15	2.49 ± 0.23	139.03 ± 7.09
1000	97.93 ± 0.70	54.15 ± 1.16	33.44 ± 0.44	22.56 ± 1.56
2500	542.61 ± 12.26	285.28 ± 2.75	161.27 ± 0.99	94.03 ± 2.54
5000	3697.97 ± 203.33	1082.12 ± 8.28	574.06 ± 8.13	330.40 ± 7.01

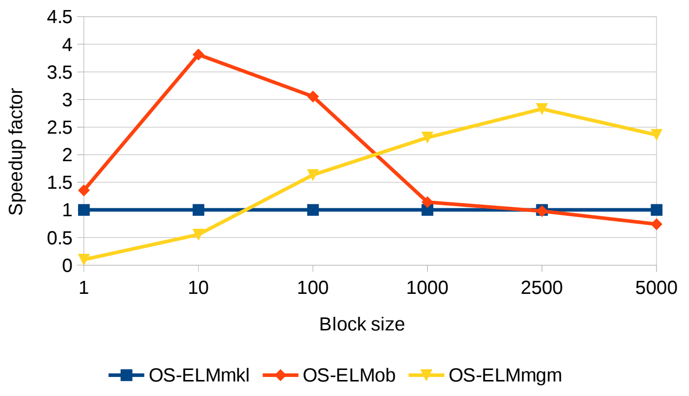


Figure 1: Speedups of TRTs on GPU Cluster

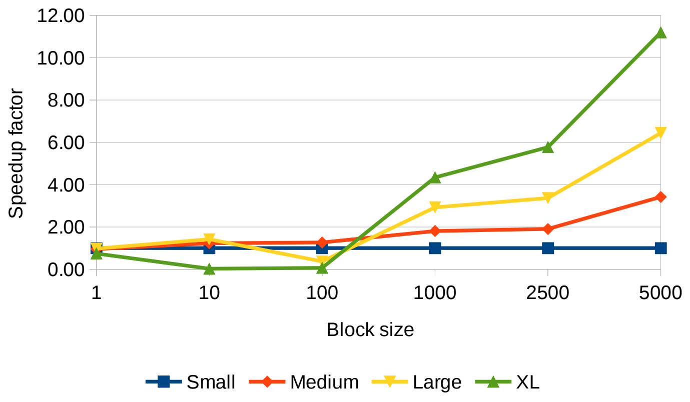


Figure 2: Speedups of OS-ELMmkl on VMs

with one thread, in Medium VM with two threads, in Large VM with four threads and in XL VM with 8 threads.

Observing table 4 and figure 2, we can notice that, in general, OS-ELMmkl reduced the TRT according to the increase in VM size. The exceptions are all cases with BS=1, which presented an inverse pattern (the higher the VM, the higher the TRT), BS=100 for Large and XL VMs, and BS=10 for XL VM, the last two being the most abnormal.

5 Conclusions

The results showed that BS is determinant for the TRTs of OS-ELMob, OS-ELMmkl, and OS-ELMmgm. Cases with BSs of up to 100 instances gave better results when implemented with OpenBLAS and Intel MKL. We can infer that this behavior relates to the memory communication overload between CPU and GPU in the MAGMA functions.

In BSs of up to 100 instances, some matrix operations are far less complex in the OS-ELM kernel. On the other hand, this generates a higher number of iterations at its kernel, which increases the number of function calls (many calls to operations with little complexity). That approach makes this overhead time considerable, making use of the hybrid functions of MAGMA disadvantageous.

From the execution in VMs, we can note that, in general, OS-ELMmkl reduced the TRT according to the increase in VM size. The above-mentioned abnormal cases need to be better studied, but one point to note is that in the GPU cluster, both OS-ELMmkl and OS-ELMob did not take as long with BS=10 and BS=100, even running with 16 threads.

In this work, we get significant results related to TRT, so the next steps have constructed an ensemble that distributes several models with one of the developed implementations as kernels, functioning as operators of a Complex Event Processing System, with the intention to improve the RMSE.

Acknowledgments

The authors would like to acknowledge the HPI Future SOC Lab for providing the resources used in this work and the support of NVIDIA Corporation with the donation of the GPU used for initial tests of this research. The first author would also like to acknowledge the Federal Institute of Education, Science and Technology of São Paulo at Piracicaba for supporting his research.

References

[1] A. Akusok, K. M. Bjork, Y. Miche, and A. Lendasse. "High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications". In: *IEEE Access* 3 (2015), pages 1011–1025. DOI: 10.1109/ACCESS.2015.2450498.

[2] A. Bueno, G. P. Coelho, and J. R. Bertini. "Online Sequential Learning based on Extreme Learning Machines for Particulate Matter Forecasting". In: *Brazilian Conference on Intelligent Systems (BRACIS)*. 2017, pages 169–174. DOI: 10.1109/BRACIS.2017.25.

[3] CETESB. *Qualidade do ar no estado de Sao Paulo 2015*. Tech. Rep. 2016. URL: http://cetesb.sp.gov.br/publicacoes-relatorios/ (last accessed 2016-12-31).

[4] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. "A survey on concept drift adaptation". In: *Computing Surveys* 46.4 (2014), pages 1–37. DOI: 10.1145/2523813.

[5] J. Honaker, G. King, and M. Blackwell. "Amelia II: A Program for Missing Data". In: *Journal of Statistical Software, Articles* 45.7 (2011), pages 1–47. ISSN: 1548-7660. DOI: 10.18637/jss.v045.i07.

[6] G.-B. Huang. *MATLAB Codes of ELM Algorithm*. 2013. URL: http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html (last accessed 2016-12-31).

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70.1 (2006). Neural Networks, pages 489–501. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2005.12.126.

[8] Intel. *Intel Math Kernel Library*. 2017. URL: https://software.intel.com/mkl (last accessed 2016-12-31).

[9] B. Krawczyk. "GPU-accelerated extreme learning machines for imbalanced data streams with Concept Drift". In: *Procedia Computer Science* 80 (2016), pages 1692–1701. DOI: 10.1016/j.procs.2016.05.509.

[10] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararaja. "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks". In: *IEEE Transactions on Neural Networks* 17.6 (2006), pages 1411–1423.

[11] NVIDIA. *CUDA Toolkit 4. 2 CUBLAS Library*. Feb. 2012. URL: https://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf (last accessed 2016-12-31).

[12] R. Souza, G. P. Coelho, A. E. A. Silva, and S. A. Pozza. "Using Ensembles of Artificial Neural Networks to Improve PM10 Forecasts". In: *Chemical Engineering Transactions* 43 (2015), pages 2161–2166.

[13] S. Tomov, J. Dongarra, and M. Baboulin. "Towards dense linear algebra for hybrid GPU accelerated manycore systems". In: *Parallel Computing* 36.5-6 (June 2010), pages 232–240. DOI: 10.1016/j.parco.2009.12.005.

[14] C. M. Vong, W. F. Ip, P. K. Wong, and C. C. Chiu. "Predicting minority class for suspended particulate matters level by extreme learning machine". In: *Neurocomputing* 128 (2014), pages 136–144. DOI: 10.1016/j.neucom.2012.11.056.

[15] X. Zhang, Q. Wang, and Y. Zhang. "Model-driven level 3 BLAS performance optimization on Loongson 3A processor". In: *Proceedings of the International Conference on Parallel and Distributed Systems ICPADS*. 2012, pages 684–691.

BPChain: A Benchmarking Tool for Comparative Evaluation of Blockchain Protocols

Jonas Cremerius, Simon Siegert, Anton von Weltzien, Annika Baldi, Finn Klessascheck, Svitlana Kalancha, and Tom Lichtenstein

Hasso Plattner Institute, University of Potsdam, Germany
{first.last}@student.hpi.uni-potsdam.de

Blockchain technology, promising to transform traditional business networks, quickly attracted attention of enterprises. In response, many blockchains protocols were released offering different benefits for business. Thus, it significantly complicated the choice of the technology for blockchain powered applications . However, so far no universal comparison framework, which provides objective measures of different protocols, exists. Aiming to perform a comparative evaluation of blockchain protocols we are working on a benchmarking tool, which will give users the possibility to choose the best fit for a concrete scenario. The platform provides a real-time visualisation of metrics describing the blockchains' performance. Furthermore, the platform will allow to run scenarios on configured blockchains, which will help users to choose the right blockchain protocol for a specific case. This paper presents our preliminary results, which includes the approach, the architecture, and first benchmarking results.

1 Introduction and Motivation

Blockchain based on distributed ledger technology allows to optimize business operations and make them more transparent, thus, enabling trustless interactions between participants. Many applications using this evolving technology are now available, and in response, several blockchain platforms have been developed. These platforms have different characteristics and offer various capabilities empowering users to create their own blockchain applications. The challenge is to choose the right platform for a specific use case. The benchmarking tool should meet following requirements to provide a qualitative analysis:

- Provide support of different blockchain technologies.
- Identify specific and independent metrics for evaluation of performance of blockchain protocols.
- Allow configuration and setup of blockchain parameters.
- Run different scenarios in blockchain networks to compare their performance for a concrete use case.

In general, blockchain platforms can be classified into two main types: permissionless and permissioned. Permissionless systems such as Ethereum [5] or Bitcoin [3] are publicly available for use. Even if the network is configured as private (in the case of Ethereum), each participating node can still conduct transactions as well as take part in the consensus process to run the blockchain. Permissioned platforms, such as MultiChain or Hyperledger, allow to build a so called consortium, where every new node must be approved before joining the network and conducting transactions. To discover strengths and weaknesses of both blockchain network types, we deployed and tested Ethereum, XAIN, and MultiChain systems.

- Ethereum is a general purpose blockchain platform that supports smart contracts and is open and permissionless. It presently uses its own Proof of Work (PoW) consensus protocol [5].
- MultiChain is a permissioned platform for the creation and deployment of private blockchains, either within or between organizations. MultiChain utilizes Byzantine Fault Tolerance (BFT) consensus protocol and does not include support for smart contracts [2].
- XAIN (eXpandable Artificial Intelligence Network) is a framework for the development of intelligent and adaptive blockchain platforms. The XAIN framework supports a new consensus protocol, Practical Proof of Kernel Work (PPoKW), a variant of Proof of Work based on Cryptographic Sortition. XAIN blockchain supports smart contracts [8].

2 Related work

The latency of blockchain based systems has already been investigated by Yasaweerasinghelage et al. [9]. In the paper they investigated the feasibility of using architectural performance modelling and simulation tools to predict the latency of blockchain-based systems. In another work, Weber researched the question of read and write availability in transaction management, comparing Ethereum and Bitcoin networks [7]. Furthermore we looked into the project of National Singapore University, which aims to create a framework for analyzing private blockchains. The project focused on the evaluation of three private blockchains: Ethereum, Parity, and Hyperledger Fabric. The research concluded that these systems are still far from displacing current database systems due to gaps in performance [1]. The review of related work shows that the evaluation framework for blockchains is researched actively and that further evolvement of the technology plays a big role for enterprises.

3 BPChain: solution for blockchains benchmarking

3.1 Estimation criteria and blockchain metrics

Each tested technology has strengths and weaknesses, depending on the underlying consensus algorithm and network configuration. The choice of platform is determined on overall efficiency, scalability, security and maturity of its blockchain, which can be evaluated through the network parameters.

Scalability of the blockchain network is its ability to reach consensus when the number of peering nodes are constantly increasing. Networks using PoW have significant problems with scalability: more peers in the network and therefore more transactions result in longer time to reach consensus and to create a block. If the block size parameter is set too low, more blocks will be needed to cover transactions resulting in longer consensus time [6]. Therefore, generally an Ethereum chain is difficult to scale, while MultiChain does not experience scalability problems.

Efficiency of blockchain is measured against two criteria: speed and cost of transactions. Ethereum demands a certain fee to conduct a transaction, whose size grows with the number of nodes in the network. The geography of miners as well as their computing power define the throughput limit of the blockchain, thus the speed of transactions confirmation. Blockchain protocols operating PoW have a lower transaction throughput, because each block of valid transactions must be verified and stored in the chain, demanding high computation power. BFT-based blockchains, such as MultiChain, operate without mining and transaction fees ensuring higher transaction throughput.

Security of blockchain relies on robustness against external and internal attacks. While most implementations make the data itself immutable, that is not the same as secure. Ethereum has experienced sufficient hacks over the past years, meaning it is difficult to consider it secure by default. On the contrary, Ethereum is more stable against internal attacks, excluding the possibility for a majority of nodes to unite and and sophisticate the network, while such an internal attack is possible in MultiChain.

3.2 Architecture

Our benchmarking tool (BPChain) consists of a web interface, a backend server, and a blockchain layer. The web interface communicates with the server requesting the information from chosen blockchains and showing the benchmarking results to the user. The backend server consists of a database for storage and aggregation of blockchain data and controllers to ensure communication between frontend and blockchain layer. The Chain data collector receives information from Nanopool [4], which provides statistics from public Ethereum network, as well as from nodes, running in private chains on FSOC servers. Users can configure private networks, which is handled by Private Chain configurator component. It communicates with FSOC through WebSockets. Both frontend and backend are running on a server as Docker component.

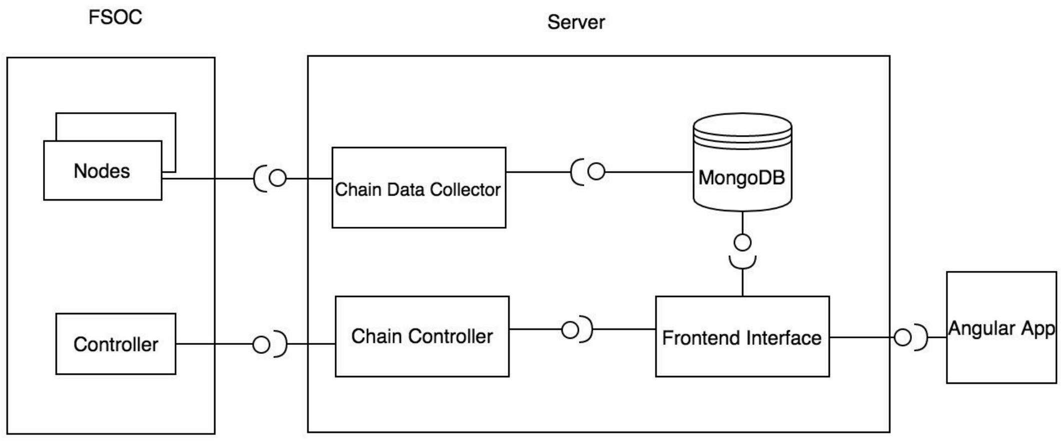


Figure 1: Solution architecture

3.3 Test scenario: EVAPCoin

Considering the blockchain evaluation parameters, we modelled a use case to test the selected blockchain protocols: each student of HPI being enrolled in lectures and seminars can evaluate them and receive so called HPI coins in return. As a reward, the coins can be spent at the HPI events for beverages and food or services. Besides each student participating in any university club can be rewarded with a certain amount of coins for supporting clubs activities. The current process of coins distribution is not transparent because students cannot retrace the number of earned and spent coins. Therefore, we modelled the following scenario: each student has a registered account on the blockchain to manage their own coins, and all transactions such as creating, sending, or spending coins are stored on the blockchain. This ensures a transparent and non-fraudulent distribution of coins.

3.4 Used FSOC resources

All experiments were executed on a Fujitsu RX600S5-2 architecture with 4 Xeon X7550 processors, each having eight cores with 2 GHz, providing 1024 GB of RAM and running Java 7 and Ubuntu 14.04. In the first phase of our project we built private Ethereum, XAIN, and MultiChain networks, which are deployed to the FSOC datacenter. Every node of each network runs in a separate Docker container, which allows to scale the network easily by incrementing the number of Docker containers. The provided resources make it possible to compare the performance of networks running the above-described scenario.

3.5 Findings

The derived results are based on the implementation of the defined test scenario. Both Ethereum and XAIN run smart contracts, which simplifies the implementation of the chosen use case. MultiChain does not support smart contracts but allows to send assets between network participants. As the scenario does not demand complex logic and automation, the capabilities of MultiChain are sufficient to deploy it.

All researched blockchains were capable to execute the scenario, but the experiment produced different results for each network. Starting with 10 nodes, we were constantly increasing the number of nodes to test scalability and efficiency of networks. There are no observable results in the ten-node network: each blockchain handled it easily. Increasing the number of nodes to 100 resulted in a capacity overload in the Ethereum network. The speed of transaction slowed down, while the computer resource consumption increased. XAIN and MultiChain remained stable and conducted transactions with the same speed. The next increase to 200 nodes broke the Ethereum network; the CPU was overloaded. By contrast, MultiChain still ran stable and used a reasonable amount of resources. The most interesting observation was found in the XAIN network: while both Ethereum and XAIN use PoW, the adjusted version of the protocol allowed XAIN to completely outperform the competitor network. Even though the difficulty of computation (meaning the complexity of tasks to solve for successful block creation) was set higher for XAIN, it still was much faster than the Ethereum network.

The experiment showed the inefficiency of basic PoW consensus protocol: producing abnormal consumption of computer power, the network remains unstable and unscalable. The server was already quickly overloaded by 200 nodes, which is too little for a real-world use case.

4 Conclusions

The platform delivers basic evaluation of three blockchain protocols, differentiating between public and private networks. It is possible for users to set parameters to analyze specific setups more precisely. We chose the following parameters to compare networks against similar characteristics: number of hosts and number of miners in the network, average block time, difficulty of network, and hashrate. These basic parameters give a first impression of the network performance, but still do not provide a deep analysis. Additionally, we tested the scalability of all networks by continuously increasing the number of nodes in each network. We concluded that PoW networks are harder to scale as their speed depends on the number of nodes and transactions. At the same time, networks using more resources friendly consensus protocols show better performance. FSOCs capabilities was sufficient to conduct the tests. The BPChain is a basic implementation of a benchmarking tool and meets all requirements of the first stage.

In the second stage, we are planning to advance the platform with the possibility to deploy and test smart contracts directly from the web interface. Furthermore, we are going to include other blockchain protocols with different consensus algorithms to provide users with enhanced statistics. We are working on the definition of blockchain applicability criteria to explore business scenarios and offer users a wide range of use cases.

References

[1] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan. "BLOCKBENCH". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, May 2017. DOI: 10.1145/3035918.3064033. arXiv: 1703.04057 [cs.DB].

[2] G. Greenspan. *MultiChain Private Blockchain — White Paper*. URL: https://www.multichain.com/download/MultiChain-White-Paper.pdf (last accessed 2018-04-01).

[3] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2018. URL: https://bitcoin.org/bitcoin.pdf (last accessed 2018-03-18).

[4] Nanopool. *Nanopool provides live statistics from public Ethereum*. URL: https://nanopool.org/ (last accessed 2018-04-01).

[5] R. Patel. *A next-generation smart contract and decentralized application platform*. 2018. URL: https://github.com/ethereum/wiki/wiki/White-Paper (last accessed 2018-03-18).

[6] A. Rosic. *Blockchain Scalability: When, where, how?* URL: https://blockgeeks.com/guides/blockchain-scalability/ (last accessed 2018-04-01).

[7] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A. B. Tran, and P. Rimba. "On Availability for Blockchain-Based Systems". In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, Sept. 2017. DOI: 10.1109/srds.2017.15.

[8] XAIN. *Practical Proof of Kernel Work & Distributed Adaptiveness*. URL: https://www.xain.io/pdf/XAIN-Yellow-Paper.pdf (last accessed 2018-04-01).

[9] R. Yasaweerasinghelage, M. Staples, and I. Weber. "Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation". In: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, Apr. 2017. DOI: 10.1109/icsa.2017.22.

Detecting Maximum Inclusion Dependencies without Candidate Generation*

Nuhad Shaabani and Christoph Meinel

Hasso Plattner Institute, University of Potsdam, Germany
{nuhad.shaabani,christoph.meinel}@hpi.de

Inclusion dependencies (INDs) within and across databases are an important relationship for many applications in data integration, schema (re-)design, integrity checking, or query optimization. Existing techniques for detecting all INDs need to generate IND candidates and test their validity in the given data instance. However, the major disadvantage of this approach is the exponentially growing number of data accesses in terms of the number of *SQL* queries as well as I/O operations. We introduce Mind2, a new approach for detecting n-ary INDs ($n > 1$) without any candidate generation. Mind2 implements a new characterization of the maximum INDs we developed in this paper. This characterization is based on set operations defined on certain metadata that Mind2 generates by accessing the database only 2× the number of valid unary INDs. Thus, Mind2 eliminates the exponential number of data accesses needed by existing approaches. Furthermore, the experiments show that Mind2 is significantly more scalable than hypergraph-based approaches.

*This report has appeared as
Shaabani, N., Meinel, C. (2016). Detecting Maximum Inclusion Dependencies without Candidate Generation. In: Hartmann, S., Ma, H. (eds) Database and Expert Systems Applications. DEXA 2016. Lecture Notes in Computer Science (LNCS), vol 9828. Springer, Cham. https://doi.org/10.1007/978-3-319-44406-2_10
and is thus not included here.

Exploring Game-Theoretic Formation of Realistic Networks

Tobias Friedrich, Pascal Lenzner, and David Schumann

Algorithm Engineering Group
Hasso Plattner Institute for Digital Engineering
{friedrich,pascal.lenzner}@hpi.de, david.schumann@student.hpi.de

Many real world networks from different domains share the same structural properties. So far, there only exist models which reproduce these properties via a random process and thus have only limited explanatory power. In contrast to this, we have developed an agent-based game-theoretic model which promises a better explanation of the structure of real world networks. Our new model was validated in computationally demanding large-scale experiments performed on the hardware of the HPI Future SOC Lab.

1 Introduction

Complex networks from the internet to various (online) social networks have a huge impact on our lives and it is thus an important research challenge to understand these networks and the forces that shape them. The emergence of the internet has kindled the interdisciplinary field of Network Science [3] which is devoted to analyzing and understanding real-world networks.

Extensive research, e.g. [1, 3, 4, 6, 14, 17, 19], on real world networks from many different domains like communication networks, social networks, protein-protein interaction networks, neural networks, etc. has revealed the astonishing fact that most of these networks share the following basic properties:

- *Small-world property:* The diameter and average distances in these networks are logarithmic in the number of nodes or even smaller.
- *Clustering:* Two nodes which are both adjacent to a third node have a high probability of being neighbors themselves, i.e. real networks contain an abundance of triangles and small cliques.
- *Power-law degree distribution:* In real networks the probability that a node has degree k is proportional to $k^{-\beta}$, for some constant $2 \leq \beta \leq 5$. That is, the degree distribution follows a power-law. Such networks are called *scale-free networks*.

The phenomenon that real world networks from different domains are very similar calls for a scientific explanation, i.e. formal models which generate networks with the above properties from very simple rules.

Many such models have been proposed, most prominently the preferential attachment model [4], the Chung-Lu random graph model [8], hyperbolic random graphs [13, 15] and geometric inhomogenous random graphs [5]. However, all these models describe a purely random process which eventually outputs a network having realistic properties. On the one hand, this is desirable for sampling such networks, e. g. for testing algorithms on them, but on the other hand, having a purely random process yields only a limited explanation for the structure of real world networks. Most real world networks evolved over time by the interaction of various rational agents. In case of the internet, the selfish agents correspond to the Internet Service Providers which control the Autonomous Systems; in case of social networks, the agents are people or companies who carefully choose with whom to connect. Thus, a model with higher explanatory power should consider rational selfish agents that use and modify the network to their advantage. Such models are at the core of the young field of Algorithmic Game Theory [20, 21].

2 Networks via Game Theory

In game-theoretic models for network formation, selfish agents are associated to nodes of a network. Each agent chooses as strategy any subset of other agents to form a link to. The union of all links which are chosen by some player then determines the links of the created network.

The individual goal of each agent is modeled via a cost function, which typically consists of costs for creating links and of a service cost term, which measures the perceived quality of the created network for the individual agent, e. g. the service cost could be the sum of distances to all other agents [12] or just the number of reachable agents [2].

Any assignment of strategies to agents is considered an outcome of the game. Among all those outcomes the so-called equilibria are particularly interesting. In an equilibrium no agent wants to change her current strategy, given that all other players' strategies are fixed, i. e. no agent can reduce her costs in the current situation by forming another set of links. Analyzing the structure of such equilibrium networks then ideally yields insights into why real world networks exhibit the mentioned properties.

So far, such game-theoretic approaches can explain the small-world property, that is, it has been proven that the diameter of all equilibrium networks is small [11]. However, to the best of our knowledge, no known game-theoretic model can explain the emergence of clustering and a power-law degree distribution. Thus, it is still an open problem to find and validate such a model.

3 Aims of the Project

Building on our previous work [7, 10, 16], we have developed a new game-theoretic model, called *strategic network augmentation*, which promises to solve the open problem. In initial experiments the obtained equilibrium networks from our new model have the small-world property, show significant clustering, and the node degree distribution seems to be governed by a power-law.

The aim of this project was to conduct a systematic large-scale experimental study of variants of our model. With these experiments we wanted to verify empirically that our model indeed yields equilibrium networks with all desired properties and we wanted to shed light on the influence of different parameter settings on the resulting networks. Moreover, we also wanted to compare our model with a baseline model which employs randomness instead of strategic behavior. With this, we wanted to demonstrate that in our setting strategic behavior yields a better explanation than a purely random model.

4 Used Future SOC Lab Resources

The experiments were run on the high-performance cluster of the HPI Future SOC Lab. The cluster consisted of 22 nodes with 80 cores each and 1TB of memory each[1] The experiments were run through the slurm job scheduler in batches of 75 on all nodes in parallel. Each batch took from 2 hours up to 14 days. In total we generated over 50 000 networks and extracted various properties along the way to monitor the process. Later, we analyzed the generated networks using various metrics and statistical tests.

5 Findings

The empirical study revealed that our model indeed generates networks having the small-world property, high clustering and a power-law degree distribution. A typical example of such a generated network can be found in figure 1. Figure 2 shows its degree distribution.

We could also show that the parameters of our model can be fine-tuned to adjust many of the specific properties of the generated networks. For example, we have been able to produce networks following a power-law distribution with power-law exponent $2 < \beta < 5$. This matches the observed power-law exponent of many real-world networks.

Moreover, we could demonstrate that also the average clustering coefficient [18] can be adjusted in the range from 0.2 to 0.8. This measure is the average over all nodes of their local clustering coefficient, which is the probability that two randomly chosen but distinct neighbors of some node are adjacent. In real-world networks average clustering coefficients from 0.2 to 0.7 have been observed [3, 19].

[1]Further hardware specifics can be found here: https://hpi.de/en/research/future-soc-lab/equipment.html.

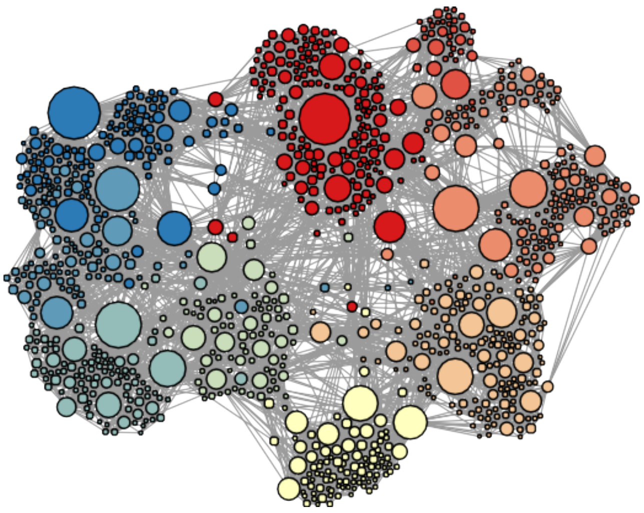


Figure 1: Generated sample network with 1000 nodes, 6840 edges, diameter 5, average path length 3.25, average clustering coefficient 0.45, colored by its 8 modularity classes. Node sizes are proportional to their degree. Drawn with Yifan-Hu layout using Gephi.

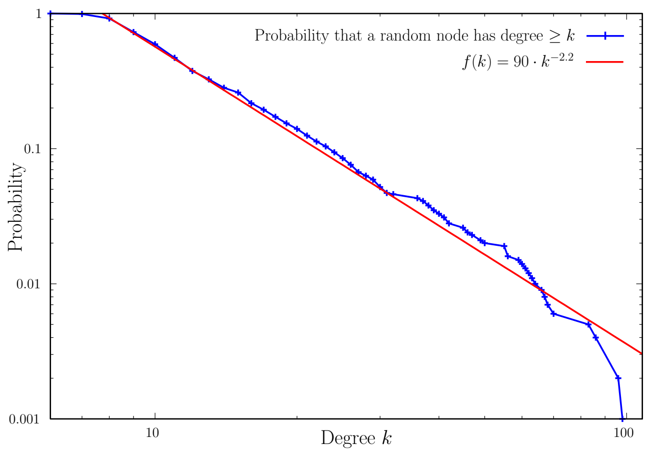


Figure 2: Log-log plot of the cumulative degree distribution of the sample network (blue) and a fitted power-law function (red). This implies that the probability $P(x)$ of a random node having degree x grows proportionally to $x^{-3.2}$.

Last but not least, our experiments showed that all our generated networks have very small diameter and average distances, i.e. that our networks are indeed small-world networks.

Another important part of our study was to compare our model with baseline models, in which some specific feature of our model was replaced. These experiments revealed that our model is indeed minimal in the sense that all core features of the models are crucially needed to produce the above mentioned results.

6 Next Steps

Besides the above mentioned properties, we have also observed other realistic properties of the equilibrium networks generated by strategic network augmentation. These need to be further investigated.

One example of an additional realistic property is the appearance of a so-called *rich club* [9], which means that the high degree nodes form a connected subgraph. Another example is that our generated networks seem to be resilient against node or edge failure, also commonly observed in real networks.

However, our previous experiments have also uncovered weaknesses of our model. First of all, the agents in our model need some global information about the network. This seems unrealistic when considering large complex networks. To address this, we want to empirically investigate truly local versions of our models. The second weakness is the lack of growth. In real networks nodes arrive (or depart) dynamically over time, which is not captured in our current model. Incorporating this feature seems to be challenging and calls for a thorough experimental study.

Last but not least, we want to validate our models with data from real networks. That is, we want to measure if real networks are close to being in equilibrium in our setting and we want to use time series of real networks to investigate if our models predict the right structural changes over time.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. "Internet: Diameter of the world-wide web". In: *nature* 401.6749 (1999), page 130.

[2] V. Bala and S. Goyal. "A noncooperative model of network formation". In: *Econometrica* 68.5 (2000), pages 1181–1229.

[3] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[4] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pages 509–512.

[5] K. Bringmann, R. Keusch, and J. Lengler. "Geometric inhomogeneous random graphs". In: *Theoretical Computer Science* 760 (2019), pages 35–54. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2018.08.014. arXiv: 1511.00576v2.

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. "Graph structure in the Web". In: *Computer Networks* 33.1 (2000), pages 309–320.

[7] A. Chauhan, P. Lenzner, A. Melnichenko, and L. Molitor. "Selfish Network Creation with Non-Uniform Edge Cost". In: *SAGT'17*. Springer. 2017, pages 160–172.

[8] F. Chung and L. Lu. "The average distances in random graphs with given expected degrees". In: *Proceedings of the National Academy of Sciences* 99.25 (2002), pages 15879–15882. DOI: 10.1073/pnas.252631999. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.252631999.

[9] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. "Detecting rich-club ordering in complex networks". In: *Nature physics* 2.2 (2006), page 110.

[10] A. Cord-Landwehr and P. Lenzner. "Network Creation Games: Think Global - Act Local". In: *MFCS'15*. Edited by G. F. Italiano, G. Pighizzini, and D. Sannella. Volume 9235. Lecture Notes in Computer Science. Springer, 2015, pages 248–260. ISBN: 978-3-662-48053-3. DOI: 10.1007/978-3-662-48054-0. URL: https://doi.org/10.1007/978-3-662-48054-0.

[11] E. D. Demaine, M. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. "The Price of Anarchy in Network Creation Games". In: *ACM Trans. Algorithms* 8.2 (Apr. 2012). ISSN: 1549-6325. DOI: 10.1145/2151171.2151176.

[12] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. "On a Network Creation Game". In: PODC'03. Boston, Massachusetts: ACM, 2003, pages 347–351.

[13] T. Friedrich and A. Krohmer. "On the diameter of hyperbolic random graphs". In: *ICALP'15*. Springer. 2015, pages 614–625.

[14] J. Kleinberg. "The Small-world Phenomenon: An Algorithmic Perspective". In: *STOC'00*. Portland, Oregon, USA: ACM, 2000, pages 163–170. ISBN: 1-58113-184-4.

[15] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. "Hyperbolic geometry of complex networks". In: *Phys. Rev. E* 82 (3 Sept. 2010), page 036106.

[16] P. Lenzner. "Greedy Selfish Network Creation". In: *WINE'12*. Edited by P. W. Goldberg. Volume 7695. Lecture Notes in Computer Science. Springer, 2012, pages 142–155. ISBN: 978-3-642-35310-9. DOI: 10.1007/978-3-642-35311-6. URL: https://doi.org/10.1007/978-3-642-35311-6.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *SIGKDD'05*. ACM. 2005, pages 177–187.

[18] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. "Random graph models of social networks". In: *Proceedings of the National Academy of Sciences* 99.suppl_1 (Feb. 2002), pages 2566–2572. DOI: 10.1073/pnas.012582999.

[19] M. Newman, A.-L. Barabasi, and D. J. Watts. *The structure and dynamics of networks*. Princeton University Press, 2011.

[20] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[21] C. H. Papadimitriou. "Algorithms, games, and the internet". In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*. Edited by J. S. Vitter, P. G. Spirakis, and M. Yannakakis. STOC'01. ACM, 2001, pages 749–753. ISBN: 1-58113-349-9.

PatientSim
Optimizing Patient Similarity Analysis

Ingmar Wiese[1], Nicole Sarna[2], Lena Wiese[3], and Araek Tashkandi[4]

[1] Research Group Knowledge Engineering
Institut für Informatik
Georg-August-Universität Göttingen
ingmar.wiese@stud.uni-goettingen.de
[2] n.sarna@stud.uni-goettingen.de
[3] wiese@cs.uni-goettingen.de
[4] araek.tashkandi@cs.uni-goettingen.de

This project focuses on a comprehensive hardware-backed analysis of biomedical data sets. The efficient implementation of such an analysis is a major precondition for large-scale evaluation of biomedical data that will be necessary in future applications in the area of personalized medicine. We applied cosine similarity to a commonly used medical dataset. Performance tests were carried out inside a HANA database instance as well as in a distributed VM environment with a machine learning toolkit.

1 Introduction

The focus of the PatientSim project lies in the area of efficient computation of patient similarity: that is, for a given target patient a doctor wants to find a group of patients (a so-called "cohort") having similar features (like for example, displaying similar symptoms or having similar lab measurements). By focusing on the cohort, a doctor can decide on an appropriate treatment based on the identified prior experiences or predict future health conditions (including mortality prediction) of the target patient. This patient similarity analysis is a major requirement for personalized medicine [9]. In general, identifying similar cohorts from electronic health records (EHRs) has been conjectured to result in a better overview of available treatment options and health predictions than the usual short-term clinical trials [8] as well as shed more light on consequences of co- and multimorbidities [1]. However, efficiently identifying cohorts with similar health conditions for a given target patient in large and diverse data sets is a major open challenge in personalized medicine.

2 Related Work

Related work on patient similarity [2, 3, 4, 10] has shown that efficiency is still a major issue. So far, similarity computations on medical data have only been done as small proof-of-concept experiments using high-level languages. For example, [8] concede that "[...] Jaccard, Dice and cosine similarities were slow [...] as they were implemented off-the-shelf and had not been optimized for speed". Only few related works analyze the optimization potential offered by modern hardware (like for example [7]). These few existing hardware-optimized implementations for the evaluation of biomedical data cover mostly small-scale fixed data sets. In our own prior work [14] we have already found out that columnar storage of patient data is advantageous.

[4] conclude that an individual patient similarity metric outperforms other statistical models that are based on patient averages. This conclusion is based on their custom developed 30-day-mortality prediction models which use cosine similarity to find the most similar patients. This improvement comes at a higher cost than traditional scores though, namely an increase in computational complexity and load. Their patient data was obtained from the MIMIC-II [5, 11, 12] database which is publicly available for researchers. It contains deidentified information that was collected from almost 30 000 ICU admissions and consists of clinical data such as vital signs every 10-15 minutes, daily lab results, and hourly urine output but also ICD-9 codes, out-of-hospital mortality data, and more. For each patient, maximum and minimum values of vital signs in 6 hour periods were extracted, as well as maximum and minimum values of lab results, which are called continuous features, and values like the ICD-9 code, called categorical features (cf. [4] for more details). These values constitute the patient vectors that are used to calculate the cosine similarity between two patients:

$$CosineSimilarity = \cos(\theta) = \frac{P_1 \cdot P_2}{\|P_1\|\|P_2\|},$$

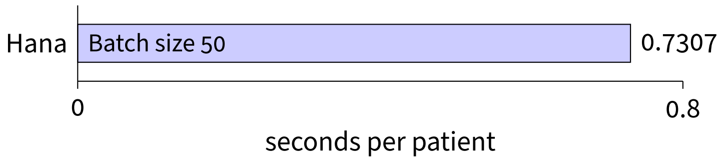
where θ is the angle between two patient vectors, $P_1 \neq P_2$. Since the cosine of a given angle is always between -1 and 1, two patients are very similar if the result is 1 and very different if the result yields -1. [4] normalized all components of the patient vectors (vital signs, etc.) to fit this range in order for them to equally contribute to the patient similarity score. All their calculations were done with *R* (version 3.1.1) while the data was obtained from a *PostgreSQL* database.

3 The Diabetes Dataset

As a test dataset we used the diabetes dataset from the Health Facts database [6, 13]. The dataset was originally extracted from the national Health Facts database (Cerner Corporation, Kansas City, MO). The database contains 10 years (1999–2008) data from EMRs of 130 hospitals throughout the United States. Different data are collected for emergency, outpatient, and inpatient visits –for instance demographic data, ICD code for diagnoses and the procedures, pharmacy data and in-hospital mortality. It consists of 74 036 643 unique visits that correspond to 17 880 231 unique patients with 117 features. The data became available for the researchers after complying with HIPAA by de-identifying all the data. It is available online by the UCI Machine Learning Repository [6]. It is called diabetes dataset since it was extracted to be used for studying the relationship between the measurement of Hemoglobin A1c (HbA1c) and early hospital readmission. Our test dataset was extracted from the original database for inpatients with any kind of diabetes as a diagnosis. Some criteria considered were the length of stay, laboratory tests and medications. 101 766 patients fulfilled these inclusion criteria.

4 Patient Similarity in HANA

In HANA the data set was imported into a column table. The norm $\|P\|$ for each patient P was calculated beforehand and stored in an additional column. The cosine similarity was run as a SQL code as shown in listing 1. We applied batch processing in the sense that the similarity for 50 patients at the same time was calculated. The runtime per individual patient is shown below. The overall computation of all pairwise similarities took roughly 20 hours which leaves room for optimization.



Listing 1: SQL code for Cosine similarity

```
SELECT p1.id, p2.id
       (p1.pid_1 * p2.pid_1 +
        ...
        p1.pid_m * p2.pi_m)
       / (p1.norm *p2.norm)
FROM   patients p1
       JOIN patients p2
       ON p1.id < p2.id;
```

5 Patient Similarity in VMs

Using the three provided FSOC VMs, the three distances among the patients of the diabetes dataset were calculated. To facilitate the computation of the distances in Java, we again utilized the jars of the Environment for Developing KDD-Applications Supported by Index-Structures (ELKI)[5] developed by the chair Database and Information Systems of the Institute for Computer Science at the University of Munich.[6]

[5]https://elki-project.github.io/ (last accessed 2018-03-15).

[6]http://www.dbs.ifi.lmu.de/research/KDD/ELKI/ (last accessed 2018-03-15).

Since the VMs had no Java version pre-installed, we decided to locally set the VMs shells to use the latest Java JDK from Oracle. Further, we used Apache's Commons CSV[7] to effectively iterate over each row of the accomplished CSV file of the diabetes dataset.

During the iteration we created for each recorded admission a patient vector regardless if a patient occurred more frequently in the dataset. The dimension of a patient vector is three times greater than the amount of features in the dataset, due to factorization of binary features like the gender, or features containing a specific amount of different ids like the admission type with its nine distinct values. To enable the assignment of the vectors to a specific patient, each vector was put as value and its unique admission id as key in a `Map<String, Vector>`.

After vectorizing the patients, we calculated the three distances and printed the results for each distance in its own file using the schema ⟨patientID-1, patientID-2, distance⟩. To calculate the distances between each patient vector with another efficiently, we firstly decided to let the three VMs calculate the vector distances in parallel (see figure 1) with the result that VM1 executed the distance computation between each vector of the first half of the vector map, VM2 did the same for the second half, and VM3 calculated the distances between the vectors of the first half and the vectors of the second half. Secondly, we never calculated the distance between a patient vector with itself and only executed the distance calculation for two vectors one-sided, visualized in figure 1 as white background, meaning if a we already have the distance between patientID-1 and patientID-2, we do not need to calculate it for patientID-2 and patientID-1, since it will lead to the same distance.

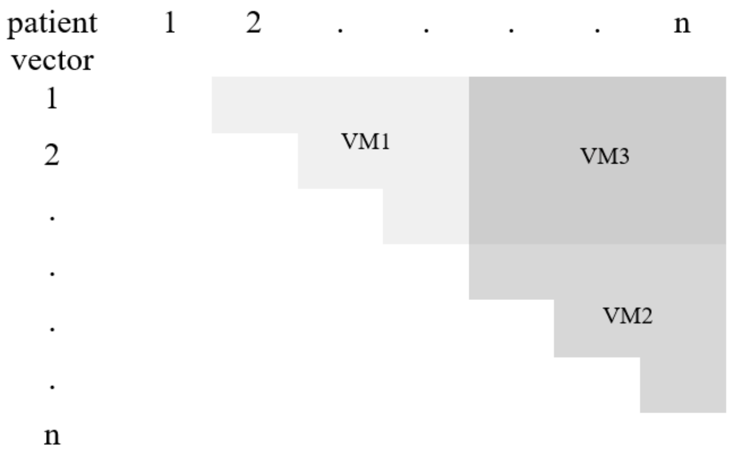
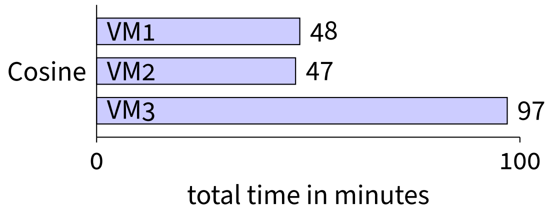


Figure 1: VMs Vector Calculation Matrix

[7] https://commons.apache.org/proper/commons-csv/ (last accessed 2018-03-15).

We created a separate output file for each VM containing the calculated distances between the several patients. The time for calculating and writing each file was measured. The results are as follows:



6 Conclusion and Future Work

Computing the cosine similarity for all pairs of patients in our data set was currently significantly faster with the machine learning tool on the three VMs than with the single HANA instance. We plan to investigate more on potential optimizations for the HANA system. Based on the computed similarities, we plan to train and test prediction models for personalized medicine in the future.

References

[1] B. Gallego, S. R. Walter, R. O. Day, A. G. Dunn, V. Sivaraman, N. Shah, C. A. Longhurst, and E. Coiera. "Bringing cohort studies to the bedside: framework for a 'green button'to support clinical decision-making". In: *Journal of comparative effectiveness research* 4.3 (2015), pages 191–197.

[2] D. Girardi, S. Wartner, G. Halmerbauer, M. Ehrenmüller, H. Kosorus, and S. Dreiseitl. "Using concept hierarchies to improve calculation of patient similarity". In: *Journal of biomedical informatics* 63 (2016), pages 66–73.

[3] A. Gottlieb, G. Y. Stein, E. Ruppin, R. B. Altman, and R. Sharan. "A method for inferring medical diagnoses from patient similarities". In: *BMC medicine* 11.1 (2013), page 194.

[4] J. Lee, D. M. Maslove, and J. A. Dubin. "Personalized mortality prediction driven by electronic medical data and a patient similarity metric". In: *PloS one* 10.5 (2015), e0127428.

[5] J. Lee, D. J. Scott, M. Villarroel, G. D. Clifford, M. Saeed, and R. G. Mark. "Open-access MIMIC-II database for intensive care research". In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE. 2011, pages 8315–8318.

[6] M. Lichman. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml. 2013.

[7] Y. Lou, A. Irimia, P. A. Vela, M. C. Chambers, J. D. Van Horn, P. M. Vespa, and A. R. Tannenbaum. "Multimodal deformable registration of traumatic brain injury MR volumes via the Bhattacharyya distance". In: *IEEE Transactions on Biomedical Engineering* 60.9 (2013), pages 2511–2520.

[8] Y. S. Low, B. Gallego, and N. H. Shah. "Comparing high-dimensional confounder control methods for rapid cohort studies from electronic health records". In: *Journal of comparative effectiveness research* 5.2 (2016), pages 179–192.

[9] J. L. Mega, M. S. Sabatine, and E. M. Antman. "Population and personalized medicine in the modern era". In: *JAMA* 312.19 (2014), pages 1969–1970.

[10] M. Panahiazar, V. Taslimitehrani, N. L. Pereira, and J. Pathak. "Using EHRs for heart failure therapy recommendation using multidimensional patient similarity analytics". In: *Studies in health technology and informatics* 210 (2015), page 369.

[11] M. Saeed, C. Lieu, G. Raber, and R. G. Mark. "MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring". In: *Computers in Cardiology*. IEEE. 2002, pages 641–644.

[12] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark. "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): a public-access intensive care unit database". In: *Critical care medicine* 39.5 (2011), page 952.

[13] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, and J. N. Clore. "Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records". In: *BioMed research international* 2014 (2014).

[14] A. Tashkandi and L. Wiese. "Leveraging Patient Similarity Analytics in Personalized Medical Decision Support System". In: *LWDA*. 2017.

A Big Data Science Experiment
Protecting Minors on Social Media Platforms

Estée van der Walt and Jan H. P. Eloff

Department of Computer Science
Security & Data Science Research Group
University of Pretoria, South Africa
estee.vanderwalt@gmail.com,eloff@cs.up.ac.za

The protection of individuals on big data platforms, like social media, is a challenge. This is largely due to the nature of these platforms allowing individuals the freedom to create and use any persona online without regulation. Victims, including minors, are exposed to various cyber threats of which identity deception is an example. In past research, various features extracted from social media platforms have been proposed to identify identity deception as found in online bot accounts. These same features were found to not have the same success when applied to the detection of deceptive online human accounts. We looked towards the field of social sciences, and more specifically psychology, to identify those features most likely to indicate lying humans. Supervised machine models were built in the hope to predict potential identity deception found for human accounts on social media platforms using these additional newly found features. For the research at hand all past results are compared to evaluate the hypothesis that these features, added through knowledge gained from psychology, improved the detection of identity deception in humans on social media platforms. Finally, an Identity Deception Score (IDS) is presented in the hope to further explain why specific individuals are perceived as being potentially deceptive.

1 Project idea

Social media allows individuals to add content at will [8]. More data does, however, create more cyber threats [9]. The volume of data alone makes it difficult to monitor each online social media account. This volume, combined with data being added at great speed and in different formats, like video and image, adds to the challenge of protecting individuals against a plethora of cyber threats [10].

One such cyber threat is identity deception [14]. This is when a human lies about who they are. In a social media context, this is when an online social media account's information (human or bot) is not correlated with the truth about the actual account holder [5]. An example would be a man in his 40s describing himself as a 14-year-old boy on Facebook.

Past research has tried to address this problem by finding such deceptive accounts generated by bots [1, 3, 12]. There are more examples of such fake accounts which allows for more research opportunity. Past research has also proposed various new features that could indicate deception, like the friend/follower ratio on Twitter [2]. No research has been found to date, presenting features, new or engineered, given what we have learnt about the deceptive nature of humans from other research fields like the social sciences.

Within the field of social sciences, and specifically psychology, much research has been done in trying to understand why humans lie. Not only were the humans' motive for lying investigated amongst others [4], but also what they lie about [16]. For this research we have focused on what humans have been found to lie about and what similar features are available or could be engineered on social media platforms. The hope is that these additional engineered features will aid in the detection of identity deception of humans on social media platforms.

Various research experiments have been executed to evaluate the hypothesis that these features from psychology, aid in the more successful detection of identity deception. The research project has been divided into various steps discussed in more detail during previous research papers [17] and follows a scientific approach. The focus of this phase of the research, highlighted in figure 1, was to evaluate the results from all previous experiments. The results are discussed in section 3 of this report. Lastly this phase also proposes an Identity Deception Score (IDS) to further explain why one human is perceived to be deceptive and another not.

Main deliverables

The main deliverables of the past six months were:

- To combine all results from previous experiments.
- To compare the performance of all machine learning algorithms used across all experiments.
- To create an IDS. The IDS is supposed to indicate whether an account is deceptive or not and should also include reasoning as to why that decision was reached.

2 Use of HPI Future SOC Lab resources

To reiterate past feedback, the following resources were used for the research at the HPI Future SOC lab:

- Twitter: The Twitter4j Java API was used to dump the data needed for the experiment in a big data repository [15].
- Hortonworks Hadoop 2.4 [6]: For the purposes of this experiment HDP Hadoop runs on an Ubuntu Linux virtual machine hosted at the HPI Future SOC Lab in Potsdam, Germany. This machine contains 4 TB of storage, 8 GB

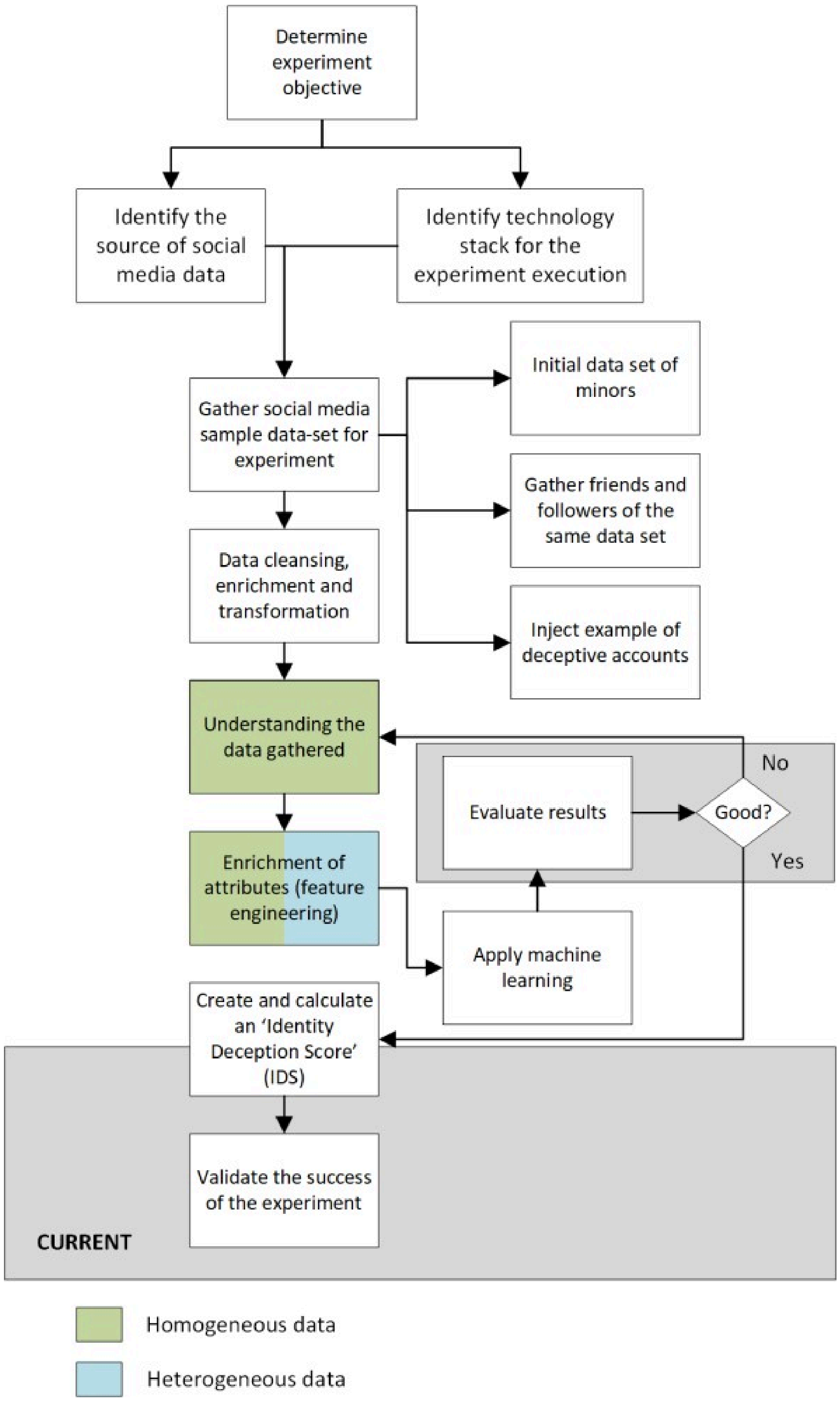

Determine experiment objective
Identify the source of social media data
Identify technology stack for the experiment execution
Gather social media sample data-set for experiment
Initial data set of minors
Gather friends and followers of the same data set
Inject example of deceptive accounts
Data cleansing, enrichment and transformation
Understanding the data gathered
Evaluate results
Good?
No
Yes
Enrichment of attributes (feature engineering)
Apply machine learning
Create and calculate an 'Identity Deception Score' (IDS)
Validate the success of the experiment
CURRENT
Homogeneous data
Heterogeneous data

Figure 1: The project process diagram

RAM, 4 ×Intel Xeon CPU E5-2620 @2 GHz and 2 cores per CPU. Hadoop is well known for handling heterogeneous data in a low-cost distributed environment, which is a requirement for the experiment at hand.

- Flume: Flume is used as one of the services offered in Hadoop to stream initial Twitter data into Hadoop and into SAP HANA.
- Ambari: For administration of the Hadoop instance and starting/stopping the services like Flume.
- Java: Java is used to enrich the Twitter stream with additional information required for the experiment at hand and automate the data gathering process.
- SAP HANA [13]: A SAP HANA instance is used which is hosted at the HPI Future SOC Lab on a SUSE Linux operating system. The machine contains 4 TB of storage, 2 TB of RAM (1.4 TB effective) and 96 cores. The in-memory high-performance processing capabilities of SAP HANA enables almost instantaneous results for analytics.
  The XS Engine from SAP HANA is used to accept streamed Tweets and populate the appropriate database tables.
- Machine learning APIs: Various tools are considered to perform classification, analysis and apply deep learning techniques on the data. These include the PAL library from SAP HANA, SciPy libraries in Python, Spark Mlib on Hadoop, and the Hadoop Mahout service. For the research, R was the final choice. This decision was made due to support on this platform and libraries freely being available on the web community at a large scale.
- An additional Linux machine was provided for the lab to aid in the running of the CPU and memory intensive machine learning algorithms. The VM has 8 cores and 64 GB of RAM.
- Visualization of the results will be performed by the libraries in R [7] and PowerBI [11] where appropriate.

The following ancillary tools were used as part of the experiment:

- For connection to the FSOC Lab we used the OpenVPN GUI as suggested by the lab.
- For connecting and configuration of the Linux VM instance we used Putty and WinSCP
- For connecting to the SAP HANA instance, we used SAP HANA Studio (Eclipse) 1.80.3

3 Findings in the Fall 2017 semester

For this phase of the research, the results from previous supervised machine learning results were compared against each other. This is done in the hope of understanding which feature sets were best at identifying deceptive online humans. This evaluation was compared against the hypothesis that engineered features, borrowed from the field of psychology, can significantly increase the successful detection of identity deception.

The three experiments were as follows:

- The first experiment used attributes found in social media alone as the features to identify deceptive online accounts.
- The second experiment looked towards past research done in the detection of bot accounts to add more engineered features in the supervised model trained to detect deceptive online accounts.
- The last experiment engineered new features given past research from psychology on why and what humans lie about. These new features were added in hope of training an even better supervised machine learning model.

The results from all three experiments are presented in table 1. It shows how each successive experiment improved on the results from the previous one.

Table 1: Comparative results from experiments

Machine learning algorithm	Exp1 (Meta)	Exp2 (Bots)	Exp3 (Psychology)
svmRadial	15.09 %	32.16 %	89.18 %
rf	31.77 %	49.75 %	96.15 %
J48	27.16 %	44.53 %	91.64 %
bayesglm	15.33 %	33.41 %	87.84 %
knn	22.62 %	40.43 %	89.06 %
Adaboost	29.24 %	47.54 %	94.44 %
rpart	23.05 %	32.37 %	91.56 %
nnet	27.90 %	41.07 %	89.43 %

Lastly, an IDS was produced. The IDS predict the potential of a human being deceptive. An example of such a scenario would be once the above machine learning model is applied, it produced only a prediction of 84 % that person X is deceptive. This information is however insufficient to explain why this decision was reached. The IDS proposes to also give additional information as to what features contributed to the 84 % and how much. An example of one IDS is shown in table 2.

With this information it is now possible to understand why a decision was reached.

Table 2: IDS contribution example result

Feature	Contribution
FOLLOWERS_COUNT	-0.12 %
FRIENDS_COUNT	-0.09 %
FF_RATIO	-0.01 %
LISTED_COUNT	0.01 %
USERNAME_LENGTH	12.84 %
GEO_ENABLED	0.03 %
PROFILE_HAS_URL	-0.43 %
ACCOUNT_AGE_IN_MONTHS	1.84 %
HAS_NAME	23.51 %
HAS_IMAGE	0.05 %
DUP_PROFILE	0.88 %
HAS_PROFILE	0.08 %
STATUS_COUNT	0.13 %
DISTANCE_LOCATION	0.09 %
DISTANCE_TZ	3.59 %
COMPARE_GENDER	-0.56 %
LEVENSHTEIN	24.51 %
COMPARE_AGE	33.66 %

4 Architecture

The SAP HANA instance, virtual machines and storage was provided by the HPI FSOC research lab and the following is worth mentioning:

- There were no issues in connection.
- The lab was always responsive and helpful in handling any queries.
- The environment is very powerful, and more than enough resources are available, which makes the HPI FSOC research lab facilities ideal for the experiment at hand.
- Without the additional VM with more cores, we would not have been able to perform the machine learning computations.

Overall, we found that the environment and its power enabled the collection and handling of a big dataset without issue. The support of the HPI FSOC research lab is greatly appreciated.

5 Next steps for 2018

The deliverables for this next phase are as follow:

- To explain the IDS results in more detail,
- Investigate whether the IDS could lead us to understand which features online are most indicative of deception.

References

[1] B. van den Belt. *How to recognize Twitter bots: 7 signals to look out for*. Aug. 20, 2012. URL: https://www.stateofdigital.com/how-to-recognize-twitter-bots-6-signals-to-look-out-for/.

[2] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. "Fame for sale: efficient detection of fake Twitter followers". In: *Decision Support Systems* 80 (2015), pages 56–71. DOI: 10.1016/j.dss.2015.09.003.

[3] J. P. Dickerson, V. Kagan, and V. Subrahmanian. "Using sentiment to detect bots on Twitter: Are humans more opinionated than bots?" In: *Advances in Social Networks Analysis and Mining (ASONAM)*. 2014, pages 620–627. DOI: 10.1109/ASONAM.2014.6921650.

[4] M. Drouin, D. Miller, S. M. Wehle, and E. Hernandez. "Why do people lie online? "Because everyone lies on the internet"". In: *Computers in Human Behavior* 64 (2016), pages 134–142. ISSN: 0747-5632. DOI: 10.1016/j.chb.2016.06.052.

[5] J. T. Hancock and C. L. Toma. "Putting your best face forward: The accuracy of online dating photographs," in: *Journal of Communication* 59 (2009), pages 367–386. DOI: 10.1111/j.1460-2466.2009.01420.x.

[6] Hortonworks. (*2014*). URL: http://hortonworks.com/partner/sap/ (last accessed 2017-03-15).

[7] R. Ihaka and R. Gentleman. "R: A Language for Data Analysis and Graphics". In: *Journal of Computational and Graphical Statistics* 5.3 (1996), pages 299–314. DOI: 10.1080/10618600.1996.10474713.

[8] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money. *Big Data: Issues and Challenges Moving Forward,* 2013, pages 995–1004. DOI: 10.1109/HICSS.2013.645.

[9] R. P. Khandpur, T. Ji, S. Jan, G. Wang, C.-T. Lu, and N. Ramakrishnan. "Crowdsourcing Cybersecurity. Cyber Attack Detection using Social Media". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, Nov. 2017. DOI: 10.1145/3132847.3132866. arXiv: 1702:07745 [cs.CR].

[10] H. Lipson. *Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues*. Technical report CMU/SEI-2002-SR-009. Carnegie Mellon University, Nov. 2002. DOI: 10.1184/R1/6585395.v1.

[11] Microsoft. *Microsoft Power BI*. URL: https://powerbi.microsoft.com/ (last accessed 2017-08-15).

[12] B. Rashidi and C. Fung. "BotTracer: Bot user detection using clustering method in RecDroid," in: *Network Operations and Management Symposium (NOMS)*. 2016, pages 1239–1244. DOI: 10.1109/NOMS.2016.7502994.

[13] SAP SE. *SAP HANA, Enterprise Edititon*. URL: https://www.sap.com/products/hana.html (last accessed 2017-08-15).

[14] M. Tsikerdekis and S. Zeadally. "Detecting and Preventing Online Identity Deception in Social Networking Services," in: *Internet Computing, IEEE* 19 (2015), pages 41–49. DOI: 10.1109/MIC.2015.21.

[15] Twitter. *Twitter API*. URL: https://dev.twitter.com/overview/api (last accessed 2017-03-15).

[16] S. Utz. "Types of deception and underlying motivation: What people think," in: *Social Science Computer Review* 23 (2005), pages 49–56. DOI: 10.1177/0894439304271534.

[17] E. V. der Walt and J. H. P. Eloff. "Protecting minors on social media platforms - A Big Data Science experiment". Presented at the HPI Cloud Symposium "Operating the Cloud", Potsdam, Germany. 2015.

A GPU-Accelerated Skeleton Discovery for Gaussian Distribution Models

Christopher Schmidt, Johannes Hügle, Siegfried Horschig, and Matthias Uflacker

Enterprise Platform and Integration Concepts
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

The estimation of causal graphical models allows to solve important problems in many different domains. For example in genetics, gene regulatory networks can be seen as a practical embodiment of systems biology and can be used for drug design or diagnostics. There are several constraint-based algorithms for estimating these causal graphical models, all sharing the skeleton discovery of the well-known PC algorithm. This part of the algorithm, known as skeleton discovery, has a high computational complexity, which increases exponential to the number of variables. In our work, we investigate the use of Graphics Processing Units (GPUs) to address this challenge and speed up the algorithm's execution time. We propose a first version of a fully CUDA-based implementation for the skeleton discovery and provide an evaluation based on real world gene expression data. The results indicate a benefit, but also show limitations in our current version. For these we propose strategies to be addressed in future work.

1 Introduction

Understanding causal relationships between observed variables in complex systems enables new insights and is of particular interest in high-dimensional settings such as genetic research. In this context, causal relationships are depicted as a causal graphical model where directed edges between vertices represent direct causal relationship between observed variables [19]. For example in genetic research, the construction of gene regulatory networks inferred from gene expression data allows for solving a number of different biological and biomedical problems, e.g., for drug design or diagnostics [22].

Constraint-based algorithms for learning these causal structures, e.g., the PC algorithm introduced by Spirtes et al. [26], use conditional independence (CI) tests to receive information about underlying relationships. Building on this skeleton, the algorithm determines the orientation of the detected undirected relationships to construct a causal graphical model. Besides the PC algorithm, there exist several extensions [5, 23, 26] all sharing the skeleton discovery of the PC algorithm. Hence, the improvements presented by this report can be carried out directly.

In the worst case, the computational complexity of the PC algorithm is exponential with regard to the number of variables such that the inefficiency of the algorithm hinders its application in practice, particularly for the high-dimensional gene expression datasets [11]. To address this drawback, we investigate the use of a GPU for the causal structure learning in the context of Gaussian distribution models. GPUs have been proven to be suitable execution devices for computational expensive machine learning algorithms, e.g., deep learning [9] or enterprise simulations [24]. Devices, such as the NVIDIA K80 GPU, reach a peak performance of up to 8.74 TFLOPS, are equipped with 24 GB of on-chip high bandwidth memory, and outperform Central Processing Units (CPUs) [17]. Achieving peak performance requires a data parallel task, which matches the SIMT [14] execution model of a GPU.

Our goal is to harness the parallel processing capabilities of the GPU to address the PC algorithm's computational complexity for high-dimensional datasets. In particular, we target the skeleton discovery of the PC algorithm [26] for an underlying Gaussian distribution model and develop a GPU-accelerated implementation.

The remainder of this report is organized as follows. Section 2 provides an introduction into constraint-based causal structure learning in the context of the Gaussian distribution model. In Section 3, we present our current CUDA-based implementation of the skeleton discovery. Moreover, we provide preliminary experimental results in Section 4 and discuss improvement strategies in Section 5. We close our work by providing a summary in Section 6.

2 Causal Inference Procedure

In this section, we introduce the basic concepts and the order-independent version of the PC algorithm for constraint-based causal structure learning.

2.1 Preliminaries

A causal graph is a Directed Acyclic Graph (DAG) $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ that consists of a finite set of N vertices $\mathbf{V} = (V_1, \dots, V_N)$, each representing the observed variable V_i, $i = 1, \dots, N$, and a set of directed edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ that do not contain any cycle. In this framework, a directed edge $V_i \to V_j$ of a DAG represents a direct causal relationship of V_i to V_j, for $V_i, V_j \in \mathbf{V}$. [19, 26]. It is well known, that several DAGs can describe exactly the same CI information and form a Markov equivalent class that can be uniquely described by a Complete Partially Directed Acyclic Graph (CPDAG) [2, 3].

Hence, the focus lies on the estimation of the equivalence class of the DAG $\mathcal{G}$ based on the corresponding probability distribution P of the involved variables $V_1, \dots, V_N$. In particular, under the assumption that the distribution P is generated from the true causal DAG $\mathcal{G}$ we have that there is an edge $V_i - V_j$ in the skeleton of $\mathcal{G}$ if and only if V_i, V_j are dependent given all $\mathbf{S} \subseteq \mathbf{V} \setminus \{V_i, V_j\}$, see [26]. Hence, the examination of the CI information of the observed variables $V_1, \dots, V_N$ allow for the estimation of the undirected skeleton $\mathcal{C}$ of the corresponding DAG $\mathcal{G}$. The extension of the skeleton $\mathcal{C}$ to the equivalence class of the DAG $\mathcal{G}$ can be done by the repeated application of deterministic edge orientation rules on the result of the first part, e.g., see [4, 7, 25].

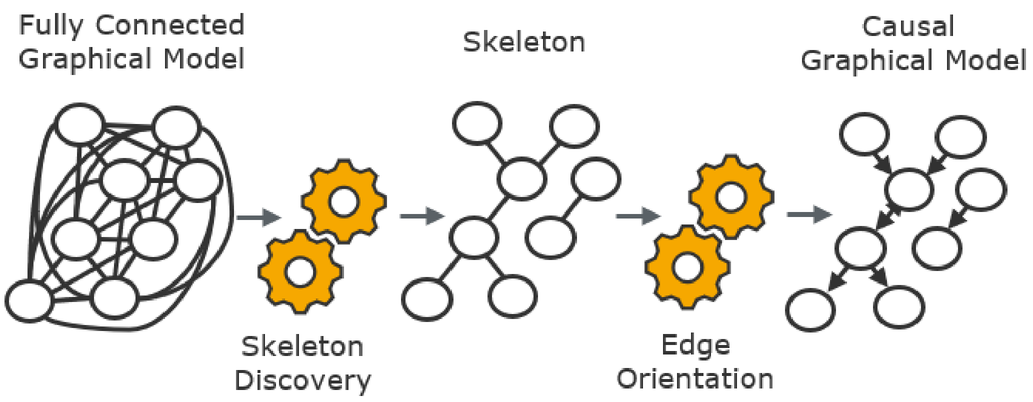


Figure 1: A schematic representation of the causal structure learning

This procedure for learning the causal graphical model was introduced by Spirtes et al. [26] and its schematic representation is depicted in Figure 1.

2.2 Gaussian Distribution Model

The appropriate CI tests executed in the skeleton discovery are directly determined by the underlying distribution of the considered variables [6]. In the context of gene expression data, involved variables $V_1, \dots, V_N$ are often assumed to be multivariate normal distributed yielding to CI tests that are based on the partial correlations [15, 22].

In this Gaussian distribution model, two variables V_i and V_j are (conditionally) independent (given $\mathbf{S} \subseteq \mathbf{V} \setminus \{V_i, V_j\}$) if and only if the (partial) correlation coefficient ρ is equal to zero [1, 10, 27]. Thus, we calculate the sample (partial) correlation coefficients based on the n observations of the involved variables $V_1, \dots, V_N$ to obtain the independence information needed in the skeleton discovery.

Note that the well-known sample correlation coefficient $\hat{\rho}$ of two variables $V_i, V_j \in \mathbf{V}$ can be used to derive the sample partial correlation coefficient via the inversion of the corresponding sample correlation matrix Cor, where $\mathrm{Cor}_{k,l}$ is the sample correlation coefficient $\hat{\rho}$ for all $V_k, V_l \in \mathbf{S} \cup \{V_i, V_j\}$ [27]. Then, the sample partial correlation coefficient can be efficiently derived via

$$\hat{\rho} = \frac{-r_{V_i,V_j}}{\sqrt{r_{V_i,V_i} r_{V_j,V_j}}}, \tag{1}$$

where r_{V_i,V_j} is the corresponding element in the inverse sample correlation matrix $\mathrm{Cor}^{-1}_{V_i,V_j}$. In order to receive a robust and accurate inverse in case of singular or higher dimensional sample correlation matrices, we compute the pseudo-inverse using the Moore-Penrose generalized matrix inverse [21].

For testing whether the (partial) correlation is zero or not, we apply standard statistical hypothesis testing theory, e. g., see [13]. Therefore, we calculate the p-value based on the corresponding sample (partial) correlation coefficient $\hat{\rho}$ via

$$\mathrm{p} = 2\left(1 - \Phi\left(\sqrt{n - |\mathbf{S}| - 3}\,|Z(\hat{\rho})|\right)\right), \tag{2}$$

where $\Phi(\cdot)$, and $Z(\cdot)$ denote the cumulative distribution function of a standard normal distribution, and the Z-transformation of the corresponding estimated (partial) correlation coefficient $\hat{\rho}$, respectively. Hence, given the significance level α, we reject the null-hypothesis $\hat{\rho} = 0$ against the two sided alternative $\hat{\rho} \neq 0$ if for the corresponding p-value it holds that $\mathrm{p} \leq \alpha$.

2.3 Constraint-Based Skeleton Discovery

Following the concepts introduced in Subsection 2.1, a naive strategy for learning the skeleton $\mathcal{C}$ would be to apply the CI test of Subsection 2.2 for all vertices $V_i, V_j \in \mathbf{V}$ given all possible subsets $\mathbf{S} \subseteq \mathbf{V} \setminus \{V_i, V_j\}$ [20]. Obviously, this would become computationally infeasible for a larger number of variables $V_1, \ldots, V_N$. A much better approach is the PC algorithm by Spirtes et al. [26] whose order-independent PC-stable version by Colombo et al. [4] is sketched in Algorithm 1. Starting with a complete undirected skeleton $\mathcal{C}$ the PC-stable algorithm uses CI-tests given an increasing separation set $\mathbf{S}$ of adjacent vertices in order to subsequently thin out the skeleton $\mathcal{C}$.

Hence, we only need to query CI tests of vertices V_i and V_j given separation sets $\mathbf{S}$ with size $l = 0$ up to the maximum size of adjacent vertices in the underlying DAG $\mathcal{G}$ (see lines 8–16 in Algorithm 1). This makes the algorithm computationally feasible even for a large number of variables, and allows for its application even in high-dimensional settings [7].

For every level l, the algorithm stores the adjacency sets $a(V_i)$ of variables V_i with respect to the current skeleton $\mathcal{C}$ (see lines 4–6). Based on this adjacency set, for every level $l = 0, \ldots, |a(V_i) \setminus \{V_j\}|$, all pairs of vertices V_i, V_j adjacent in $\mathcal{C}$ are tested for (conditional) independence (given a separation set $\mathbf{S}$ of size l) by checking if for the corresponding p-value it holds that $\mathrm{p} \leq \alpha$, or not. If the two variables V_i, V_j are independent, the edge $V_i - V_j$ is deleted from $\mathcal{C}$ and the separation set $\mathbf{S}$ is saved in **Sepset** (see lines 11–13). If all pairs of adjacent vertices V_i and V_j of the current version of the skeleton $\mathcal{C}$ are considered, the algorithm again increases l by one. This process continues until l reaches the maximum size of the adjacency sets of the vertices in the underlying DAG $\mathcal{G}$.

In order to extend $\mathcal{C}$ to the corresponding CPDAG, the derived skeleton $\mathcal{C}$ and the CI information in **Sepset** are used as basis for the application of deterministic orientation rules, e.g., see [4, 7, 25].

3 Implementation of Skeleton Discovery on GPU

In order to use the GPU to its full potential, we propose a method of level-specific kernel launches for the different levels l of the PC-stable algorithm, see Algorithm 1.

Algorithm 1: Skeleton Discovery of PC-stable [4]

```
Input: Vertex set $V$, correlation matrix Cor
Output: Skeleton $\mathcal{C}$, separation sets Sepset
Start with fully connected skeleton $\mathcal{C}$ and $l = -1$
repeat
    $l = l + 1$
    for all Vertices $V_i$ in $\mathcal{C}$ do
        Set adjacent vertices of $V_i$ in $\mathcal{C}$ by $a(V_i)$;
    end for
    repeat
        Select $V_j \in a(V_i)$ with $|a(V_i) \setminus \{V_j\}| \geq l$
        repeat
            Choose $\mathbf{S} \subseteq a(V_i)\setminus\{V_j\}$ with $|\mathbf{S}| = l$.
            if for corresponding p $\leq \alpha$ then
                Delete edge $V_i - V_j$ from $\mathcal{C}$;
                Add $\mathbf{S}$ to Sepset;
            end if
        until edge $V_i - V_j$ is deleted in $\mathcal{C}$ or
        all $\mathbf{S} \subseteq a(V_i)\setminus\{V_j\}$ with $|\mathbf{S}| = l$ are chosen
    until all adjacent $V_i$, $V_j$ in $\mathcal{C}$ such that
    $|a(V_i) \setminus \{V_j\}| \geq l$ have been considered
until all $V_i$,$V_j$ adjacent in $\mathcal{C}$ satisfy $|a(V_i)\setminus\{V_j\}| \geq l$
return $\mathcal{C}$, Sepset
```

For level $l = 0$, a test comprises two vertices $V_i, V_j \in \mathbf{V}$ with an empty separation set, i.e., $S = \emptyset$. The structure of the kernel for this call consists of a fixed amount of $\mathcal{T}$ threads for each block. The blocks are organized in a grid the size of $N \times (N/\mathcal{T})$. Thus, each thread started in this grid conducts $(N/\mathcal{T})$ CI tests, the X and Y coordinates of the block denoting V_i and V_j, respectively.

In order to test for the independency of V_i and V_j, we need to calculate the corresponding p-value via (2), see line 11 of Algorithm 1. In the implementation we use existing functions from the CUDA Math API [16], including `sqrt()` and `normcdf()`.

For level $l = 1$, CI tests are based on the corresponding sample partial correlation coefficients. As introduced in Subsection 2.2, sample partial correlation coefficients can be derived via the inversion of the corresponding sample correlation matrix Cor. As Cor is a 3×3-matrix, this can be done directly. Analogous to the previous level, we have the fixed number of $\mathcal{T}$ threads per block such that the kernel is launched with $N \times N$ blocks. Within each block all CI tests for one edge are conducted. The threads within a thread block loop through the separation set candidates in blocks of size $\mathcal{T}$. Each thread conducts one CI test and stores the result in shared memory. A master thread reduces these results and checks for an accepted CI test. If the edge is not deleted, the next block of size $\mathcal{T}$ of separation set candidates is considered for the CI tests. Potentially, the number of CI tests conducted is $N - 2$ for each possible combination of V_i and V_j. The corresponding p-value is calculated based on the sample partial correlation coefficient of V_i and V_j given $\mathbf{S}$ following (2).

For level $l \geq 2$, the CI tests get more complex as the size of $\mathbf{S}$ increases and the inverse of the sample correlation matrix cannot be calculated directly. As introduced in Subsection 2.2, for testing the CI of V_i and V_j given a separation set $\mathbf{S}$ of size $l \geq 2$ we build the corresponding sample correlation matrix Cor on the CPU. Then the partial correlations can be calculated via (1) based on the pseudoinverse of the corresponding Cor. For the calculation of the pseudoinverse via a standard Singular Value Decomposition (SVD) we use library `cuSolverDN`. After assigning a working buffer the size of $N \times N$, we call `cusolverDnSgesvd` which returns the matrices U (left singular vectors), $V^{**}T$ (right singular vectors) and the vector S (singular values). Afterwards, we construct a matrix S_{inv} with the inverse values of S aligned diagonally, the remaining fields filled with zeroes. We do so by launching a kernel with N blocks, each starting N threads. For each thread block i and the thread j the S_{inv} is filled as

$$S_{inv}(i,j) = \begin{cases} \frac{1}{S(i,j)}, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases}$$

By simply multiplying U, $V^{**}T$ and S_{inv}, we get the pseudoinverse of the corresponding sample correlation matrix. For this matrix-matrix multiplication, we use the `cuBLAS` library, specifically two calls of the `cublasSgemm` function.

The resulting matrix allows us to test CI between V_i and V_j given the set $\mathbf{S}$ by calculating the corresponding p-value based on the sample partial correlation coefficients as depicted in (2).

4 Preliminary Results

For our evaluation we conducted measurements on real world gene expression datasets [12] and compared the execution times of our implementation with an implementation from the `pcalg` R-package [8], which is executed on a CPU. The experiment is executed on two systems. One system, equipped with an NVIDIA K80 GPU, used our CUDA-based implementation. The other system, executing the `pcalg`, was equipped with an Intel i7-6700K CPU with 4 cores, hyper-threading and 64 GB of DRAM.

Table 1: Average speed-up (of 10 runs) of GPU-accelerated against CPU-based skeleton discovery on two gene expression datasets

	NCI-60	BR51
level $l = 0$	53.75	71.6
level $l = 1$	81.39	258.26
level $l = 2$	0.004	0.003
level $l = 3$	0.0002	0.0001
level $l = 4$	0.001	0.0002
Total Including Data Transfer	0.48	0.68

Table 1 shows the average speed-up in the execution times of the GPU-accelerated against the CPU-based implementation on two real world gene expression datasets. For each dataset, it presents the total speed-up, as well as the speed-up for the separate levels, during the skeleton discovery. The results show that the GPU-based implementation outperforms the implementation executed on the CPU, for level $l = 0, 1$. For levels $l \geq 2$, the GPU-based implementation is significantly slower than the CPU version. Even though, the number of tests calculated within these levels is very small compared to level $l = 0, 1$. Looking at the total runtime on GPU and on CPU, the impact of levels $l \geq 2$ mitigates the improvements achieved in level $l = 0, 1$, leading to an overall slower runtime for the GPU-accelerated version.

To understand the reasons for this bottleneck, we provide more detail to the kernel execution. For this we profiled the execution with `nvprof` [18]. The profiling of the kernel for the NCI-60 dataset shows that for level $l = 0$ and level $l = 1$ we achieve a `sm_efficiency` of almost 100%. This underlines the utilization of the parallel processing capabilities of the GPU and we think the reason for the significant speed up. The `sm_efficiency` for all kernel calls relating to levels $l \geq 2$ show a poor utilization of the Streaming Multiprocessors (SMs), with a `sm_efficiency` below 2%. We also found out that for a single test in these levels, up to 49 different kernel calls are executed. Since we cannot utilize the parallel processing power at an acceptable rate, our current implementation is slower in execution time than a

CPU-based version. Also the number of kernel calls might incur some overhead adding to the execution time.

5 Improvement Strategies for Higher Levels

In this section, we provide improvement strategies for the execution on a GPU for level $l \geq 2$. Our ideas are a result of our Evaluation in Section 4 and will be investigated in future work.

A major bottleneck in our current implementation is the low utilization of the GPU. Although we use state of the art libraries to use efficient implementations, our approach is not optimal. Given the small problem size of a single CI test, we want to investigate the capabilities of CUDA Streams[1] to process several tests at the same time on different SMs.

As another approach to overcome the current bottleneck we consider to rearrange our data in a way to allow processing several tests in a single kernel. Instead of creating small submatrices for each CI test, we construct a larger matrix placing the submatrices on the diagonal. Depending on the implementation of this approach, the required memory may become an issue.

6 Summary

In this project report, we shared our work towards a GPU-accelerated causal inference. We provided an overview on the concepts of causal inference, with a particular focus on the Gaussian distribution model. On this theoretical basis we propose an implementation of a CUDA-enabled skeleton discovery that allows to process the potentially large number of CI tests for each level in parallel. We provide some preliminary evaluation results for our first implementation, which shows significant improvements for level $l = 0$ and $l = 1$, but is limited in processing higher levels, resulting overall in similar performance to a CPU based implementation. Based on these results, we propose improvements to fully utilize the processing capabilities of the GPU to provide a more efficient version in the future.

References

[1] H. Ahrens. "Dempster, A. P.: Elements of Continuous Multivariate Analysis. Addison-Wesley Publ. Co., Reading, Mass. 1969. XII, 388 S". In: *Biometrische*

[1] http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#streams-cdp (last accessed 2018-04-01).

Zeitschrift 17.7 (1975), pages 468–468. ISSN: 1521-4037. DOI: 10.1002/bimj.19750170714.

[2] S. A. Andersson, D. Madigan, and M. D. Perlman. "A Characterization of Markov Equivalence Classes for Acyclic Digraphs". In: *The Annals of Statistics* 25.2 (1997), pages 505–541. ISSN: 00905364. URL: http://www.jstor.org/stable/2242556.

[3] D. M. Chickering. "Learning Equivalence Classes of Bayesian-network Structures". In: *J. Mach. Learn. Res.* 2 (Mar. 2002), pages 445–498. ISSN: 1532-4435. DOI: 10.1162/153244302760200696.

[4] D. Colombo and M. H. Maathuis. "Order-independent Constraint-based Causal Structure Learning". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pages 3741–3782. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2750365.

[5] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson. "Learning High-dimensional DAGs with Latent and Selection Variables". In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. UAI'11. Barcelona, Spain: AUAI Press, 2011, pages 850–850. ISBN: 978-0-9749039-7-2. URL: http://dl.acm.org/citation.cfm?id=3020548.3020648.

[6] A. P. Dawid. "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pages 1–31. ISSN: 00359246. URL: http://www.jstor.org/stable/2984718.

[7] M. Kalisch and P. Bühlmann. "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm". In: *J. Mach. Learn. Res.* 8 (May 2007), pages 613–636. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1248659.1248681.

[8] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. "Causal Inference Using Graphical Models with the R Package pcalg". In: *Journal of Statistical Software, Articles* 47.11 (2012), pages 1–26. ISSN: 1548-7660. DOI: 10.18637/jss.v047.i11.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pages 1097–1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[10] S. L. Lauritzen. *Graphical models*. Volume 17. Clarendon Press, 1996.

[11] T. Le, T. Hoang, J. Li, L. Liu, H. Liu, and S. Hu. "A fast PC algorithm for high dimensional causal discovery with multi-core PCs". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (Feb. 2015).

[12] T. D. Le, L. Liu, J. Zhang, B. Liu, and J. Li. "From miRNA regulation to miRNA-TF co-regulation: computational approaches and challenges". In: *Briefings in Bioinformatics* 16.3 (2015), pages 475–496. DOI: 10.1093/bib/bbu023.

[13] E. L. Lehmann and J. P. Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[14] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. "NVIDIA Tesla: A Unified Graphics and Computing Architecture". In: *IEEE Micro* 28.2 (Mar. 2008), pages 39–55. ISSN: 0272-1732. DOI: 10.1109/MM.2008.31.

[15] K. Marazopoulou, R. Ghosh, P. Lade, and D. Jensen. "Causal Discovery for Manufacturing Domains". In: *CoRR* abs/1605.04056 (2016). arXiv: 1605.04056.

[16] NVIDIA Corporation. *CUDA Math API*. July 2017. URL: http://docs.nvidia.com/cuda/pdf/CUDA_Math_API.pdf.

[17] NVIDIA Corporation. *NVIDIA Tesla K80 The World's fastest GPU Accelerator*. Nov. 2014.

[18] NVIDIA Corporation. *Profiler User's Guide*. Mar. 2018. URL: http://docs.nvidia.com/cuda/pdf/CUDA_Profiler_Users_Guide.pdf.

[19] J. Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 978-0-521-89560-6.

[20] J. Pearl and T. Verma. "A Theory of Inferred Causation". In: *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. KR'91. Cambridge, MA, USA: Morgan Kaufmann Publishers Inc., 1991, pages 441–452. ISBN: 1-55860-165-1. URL: http://dl.acm.org/citation.cfm?id=3087158.3087202.

[21] R. Penrose. "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Volume 51. 3. Cambridge University Press. Cambridge University Press, 1955, pages 406–413. DOI: 10.1017/S0305004100030401.

[22] A. Rau, F. Jaffrézic, and G. Nuel. "Joint estimation of causal effects from observational and intervention gene expression data". In: *BMC systems biology* 7.1 (Oct. 2013), page 111. DOI: 10.1186/1752-0509-7-111.

[23] T. Richardson. "A Discovery Algorithm for Directed Cyclic Graphs". In: *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. UAI'96. Portland, OR: Morgan Kaufmann Publishers Inc., 1996, pages 454–461. ISBN: 1-55860-412-X. URL: http://dl.acm.org/citation.cfm?id=2074284.2074338.

[24] C. Schwarz, C. Schmidt, M. Hopstock, W. Sinzig, and H. Plattner. "Efficient Calculation and Simulation of Product Cost Leveraging In-Memory Technology and Coprocessors". In: *The Sixth International Conference on Business Intelligence and Technology (BUSTECH 2016)*. 2016.

[25] P. Spirtes. "Introduction to Causal Inference". In: *J. Mach. Learn. Res.* 11 (Aug. 2010), pages 1643–1662. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1859905.

[26] P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2000.

[27] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley Publishing, 2009. ISBN: 978-0-470-74366-9.

Guilt-by-Association

Detecting Malicious Entities via Graph Mining

Pejman Najafi, Andrey Sapegin, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

In this research, we tackle the problem of detecting malicious domains and IP addresses using graph inference. In this regard, we min proxy and DNS logs to construct an undirected graph in which vertices represent domain and IP address nodes, and the edges represent relationships describing an association between those nodes. More specifically, we investigate three main relationships: subdomainOf, referredTo, and resolvedTo. We show that by providing minimal ground truth information, it is possible to estimate the marginal probability of a domain or IP node being malicious based on its association with other malicious nodes. This is achieved by adopting belief propagation, i.e. an efficient and popular inference algorithm used in probabilistic graphical models. We implement our system in Apache Spark on a large Linux machine provided by the HPI Future SOC Lab, and evaluate it using one day of proxy and DNS logs collected from a global enterprise spanning over 2 terabytes of disk space. In this regard, we show that our approach is not only efficient but also capable of achieving high detection rate (96 % TPR) with reasonably low false positive rates (8 % FPR). Furthermore, it is also capable of fixing errors in the ground truth as well as identifying previously unknown malicious domains and IP addresses.

1 Introduction

In the case of both targeted threats (e.g. social engineering, spear-phishing, Advanced Persistent Threats, etc.) and mainstream threats (e.g. drive-by download, exploit-kits, malvertising, etc.), there exists an external malicious entity administered by an adversary that successfully reaches the end client (victim). The ability to block these entities from reaching the end client is considered to be an optimum cyber security solution. That's why so many organizations heavily invest in blocking these by deploying various security solutions such as web application firewalls, proxy servers, email and web security appliance, etc. The majority of these solutions try to detect maliciousness by analyzing local features of those entities (e.g. URL structure or content of a web page). However, the problem is that the majority of these features are volatile, hence giving an advantage to cybercriminals to evade detection. On the other hand, although it is easy to change and mutate the majority of the local features, it is extremely challenging and sometimes costly to change global features (i.e. attributes shared between different malicious entities), for instance, the authoritative domain responsible for serving fast fluxing domains, the paths leading to two different landing pages, or the name server information.

In this regard, investigating the correlation between the global features of the previously known indicators of compromise (IOCs) could potentially allow us to better reason about new entities. In this paper, we formulate big data analysis for threat detection as a graph inference problem, with the intuition that malicious entities tend to have homophilic relationships with other malicious entities. More specifically, we focus on the analysis of proxy and DNS logs for the purpose of detecting malicious IP addresses and domain names based on the relationships observed in those logs and minimal prior knowledge collected from threat intelligence (TI) sources. We achieve this by adopting Loopy Belief Propagation from probabilistic graphical models which allows to propagate the labels from labeled data to unlabeled data using relationships extracted from proxy and DNS logs.

2 Our Approach

Formally we formulate our inference problem as follows:

Given: First, an undirected graph $G = (V, E)$ where V corresponds to the collection of domain names and IP addresses, and E corresponds to the set of relationships between those domain and IP nodes. V and E are extracted from events in proxy and DNS Logs. Second, binay class labels $X \in \{x_{mal}, x_{ben}\}$ defined over V, where x_{mal} represents malicious label, and $P(x_{mal})$ the probability of belonging to class malicious. Note that $P(x_{mal})$ and $P(x_{ben})$ sums to one.
Find: The marginal probability $P(X_i = x_{mal})$, i.e. the probability of node i belonging to class malicious.

In this regard, the graph $G = (V, E)$ is constructed from events in the proxy and DNS logs.

The set of vertices V consists of two types of nodes: *domain names* and *IP addresses*. Domain names are valid parts of a fully qualified domain name (FQDN) excluding the top-level domain (TLD). For instance, considering x.example.com as a given FQDN, we then take example.com as the second-level domain and x.example.com as the third-level domain. Domain names are extracted from destination URLs in proxy logs, and the query section of A records presented in DNS logs. IP addresses are validated IP version 4 addresses observed in DNS logs (A records), and occasionally in proxy logs (sometimes the URL contains an IP address rather than a FQDN, also some proxy servers log the resolved IP address).

The set of edges E expresses three distinct relationships: *subdomainOf*, *referredTo*, and *resolvedTo*. In this regard, the subDomainOf relationship captures the dependency between different levels of a FQDN, e.g. x.example.com is a subDomainOf example.com. This relationship is extracted from any valid FQDN logged in DNS or proxy logs. referredTo captures the connection between two domain/IP nodes if one has referred to the other one. This feature is extracted from the referer field in the HTTP request-header logged in proxy logs. And finally, resolvedTo captures the DNS resolution of a domain name to an IPv4 address, which is presented in DNS logs and occasionally in proxy logs. Figure 1 shows the graph constructed from raw events presented in DNS and proxy logs and illustrates different types of nodes and relationships used in graph *G*.

Proxy Log

```
2017-05-20 23:59:58 14X55 10.XXX.XXX.X 200 TCP_MISS
5XXX5 5XXX7 GET http x.example.com 80
/some/path/someFile.extension - - -
http://example2.ru/some/otherPath/X.html "Mozilla/5.0
(iPhone; CPU iPhone OS X_X_X like Mac OS X)
AppleWebKit/X (KHTML, like Gecko) Mobile/13G36 [XXX]"
- - - - - - - - - X.X.X.X - - - -
```

DNS Log

```
May 20 23:59:59 192.XXX.XXX.XXX named[XXX]: client
10.XXX.XXX.X#XXX (example.com): view 2:
query:example.com IN A + (93.184.216.34)
```

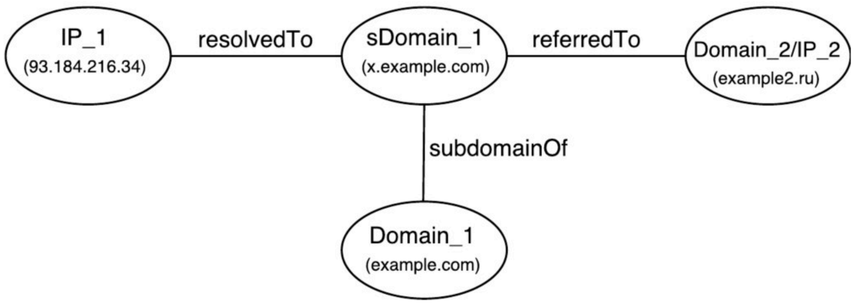


Figure 1: domain-ip graph constructed from a sample of raw events in DNS and proxy logs showing the association between domain and IP nodes using *subdomainOf*, *referredTo*, and *resolvedTo* relationships

Our intuition for these three relationships is: First, the usage of subdomains is one of the simplest yet effective techniques used by cyber criminals (e.g. DGA and domain shadowing) to evade blacklisting. Intuitively, different levels of a FQDN should belong to the same class. For instance, if x.example.com is listed as a malicious entity by a threat intelligence feed, it is likely that example.com and any other k-level domain under example.com (e.g. y.example.com) is also malicious. Second, the majority of the malware-serving networks are composed of a tree-like structure in which the victims are usually redirected through various hops before landing on the main distribution site [5]. Although different victims might land on totally different sites, the redirection paths are usually overlapped. Furthermore, the HTTP referrer is also set while a domain (e.g. a website) is loading its modules from potentially different servers, therefore, indicating association among different domains. In this regard, the HTTP referrer can be used to infer the probability of a node being malicious based on the neighboring malicious nodes that have referred to it or it has referred to. One could also expand the referrer list by implying "referring" based on the correlation among different requests presented in proxy logs (e.g. requests from the same client in a short period of time) And finally, if a domain is listed as a malicious, intuitively we could assume that the resolved IPv4 address of that domain should also be labeled malicious at least for the duration of that resolution and vice versa.

Belief Propagation Marginal probability estimation in graphs is known to be NP-complete, however, belief propagation provides a fast approximate technique to estimate marginal probabilities with time complexity and space complexity linear to the number of edges in a graph.

At the high level, BP infers a node's label from some prior knowledge about that node and other neighboring nodes by iteratively passing messages between all pairs of nodes in the graph. In this regard, in each iteration t, every node i generates its outgoing messages based on its incoming messages from neighbors in iteration $t-1$. Given that all messages are passed in every iteration, the order of passing can be arbitrary.

Table 1: Node potential based on the original state

Node	$P(malicious)$	$P(benign)$
Malicious	0.99	0.01
Benign	0.01	0.99
Unknown	0.5	0.5

Table 2: Edge potentials matrices

$\psi_{ij}(x_i, x_j)$	$x_j = benign$	$x_j = malicious$
$x_i = benign$	$0.5 + w\epsilon$	$0.5 - w\epsilon$
$x_i = benign$	$0.5 - w\epsilon$	$0.5 + w\epsilon$

Let m_{ij} denote the message sent from i to j which intuitively represents i's opinion about j's likelihood of being in state x_j. This message is a vector of messages for each possible class, i.e. $m_{ij}(x_j = malicious)$ and $m_{ij}(x_j = benign)$. Then:

$$m_{ij}(x_i) \leftarrow \sum_{x_i \in X} \phi(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{kj}(x_i) \tag{1}$$

where $N(i)$ is the set of nodes neighboring node i, and $\psi(x_i, x_j)$ is the *edge potential* which indicates the probability of a node i being in class x_i given that its neighbor j is in class x_j. $\phi(x_i)$ is called the *node potential function* which denotes the prior knowledge about a node, i.e. the prior probability of node i being in each possible class (in our case malicious and benign classes). And x_i represents a state from state space X.

The message passing phase terminates when messages do not change significantly between iterations, i.e. given a similarity threshold, the difference between the message sent from node i to node j at the iteration t and $t-1$ is less than the threshold, or when the algorithm reaches a predefined maximum number of iterations. At the end, each node will calculate its belief, which is an estimated marginal probability, or formally $b_i(x_i)(\approx P(x_i))$ which represents the likelihood of random variable X_i to take value $x_i \in \{x_{mal}, x_{ben}\}$ determined as follows:

$$b_i(x_i) = k\phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i) \tag{2}$$

where k is a normalizing constant to ensure the node's beliefs add up to 1 [7].

Node Potential Adaptation As previously explained, the node potential represents the prior knowledge about the state of each node. In this regard, we assign different node potential to domain and IP nodes based on the ground truth. For example, we assign a prior $P(X_i = x_{mal}) = 0.99$ to the node i, if i is presented in the collected malicious domain/IP list, or assigning $P(X_i = x_{mal}) = P(X_i = x_{ben}) = 0.5$ for the nodes that are neither in the malicious list nor in the benign list (i.e. they are equally likely to be malicious or benign). Note that we avoid assigning a probability of 1 to any nodes to account for possible errors in the ground truth. Table 1 shows the node potentials assigned to each vertex on the graph based on the prior knowledge (belief).

Edge Potential Adaptation We adjust the edge potential matrices to capture the intuition that neighboring nodes are more likely to have the same state due to a homophilic relationship.

Moreover, due to the fact that our graph consists of three unique edge types (referredTo, resolvedTo, subDomainOf), it is important to introduce a way to incorporate edge weight (importance). For example, two neighboring nodes that are connected via resolvedTo should influence each other more than two nodes that are connected via referredTo. This edge weight is also incorporated in the edge potential. Table 2 shows the adjusted edge potential matrices.

We experimented with different edge potential (adjusting ϵ) and although we noticed changes in final probability distributions, the end results were comparable as long as the weights captured the importance of different relationships. After experimenting with different w and E and we appointed them as follows: $w_{referredTo} = 0.5$, $w_{resolvedTo} = 1.5$, $w_{subdomainOf} = 1.5$, and $\epsilon = 0.1$. It is worth mentioning that, although the edge weights seem trivial in this research, they will have a much higher impact when adding more edge types, therefore we will investigate them further in our future work.

3 Implementation in the HPI Future SOC Lab

Due to the scalability requirements, we have implemented the BP algorithm with a parallel updating scheme using the Apache Spark framework. Apache Spark, the successor to Hadoop MapReduce is one of Apache's open-source projects that has gained so much momentum in both industry and academic due to its power of handling big data analytics. We have not only implemented the BP algorithm in Spark but also the *extract*, *transform*, *load* (ETL) modules as well as the graph itself. This design makes it possible to not only scale up (i.e. take advantage of more powerful hardware) but also scale out (i.e. distributing all modules to different machines).

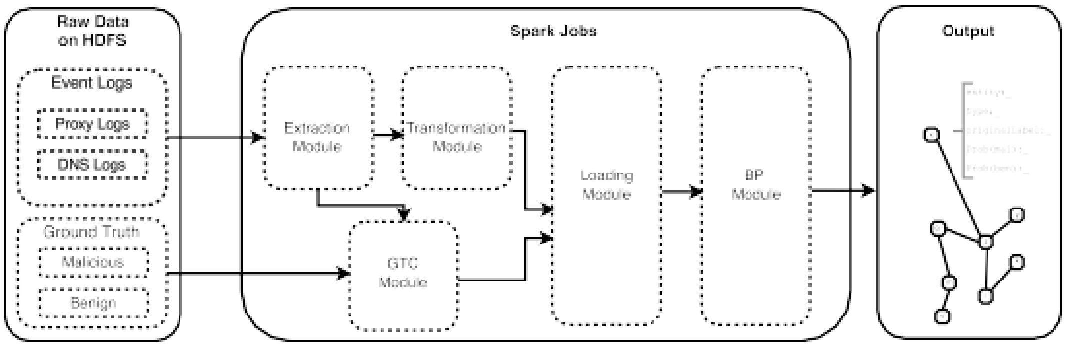


Figure 2: System Architecture Diagram

Our implemented system is composed of five modules: 1) *Extraction*, 2) *Transformation*, 3) *Ground Truth Construction* (GTC), 4) *Loading*, and 5) *BP* as shown in figure 2. In summary, the extraction module preprocesses the DNS and proxy logs by extracting, parsing, and validating the fields of interest as described in the previous sections. Then the transformation module converts the extracted values into unique vertices and edges. The ground truth construction (GTC) module is responsible for combining and adjusting the collected list of malicious/benign domain and IPv4 addresses, removing duplicate and the unmatched entities (malicious and benign entities that are in ground truth but not observed in the event logs). This module is also responsible for carefully selecting the validation set. The loading module receives the output of the transformation and GTC modules to construct a property graph and labeling each vertex based on ground truth (malicious, benign, unknown). And finally, the BP module converts the constructed graph to Markov Random Field with the provided node and edge potentials, then runs the implemented BP algorithm to compute the beliefs following the procedure described in the previous section. In order to avoid numerical underflow (zeroing-out values), the whole math performed by BP module is carried in the log domain.

Dataset description For the purpose of this research, we had access to one-day proxy and DNS logs collected from a large global enterprise. More specifically, 0.74 TB of proxy logs and 1.2 TB of DNS logs containing DNS requests and responses.

There is a total of 0.91 billion, and 0.35 billion events in DNS and proxy logs, respectively. After running ETL modules on those events we were capable of extracting approximately 1.89 million unique vertices and 4.29 million unique edges.

Ground Truth description To assign priors to domain and IP vertices, a ground truth set was prepared by collecting a list of known malicious domain and IP addresses from both a commercial threat intelligence platform and various freely available sources including (but not limited) to Google Safe Browsing, AlienVault Open Threat Exchange, malwaredomainlist.com, malwaredomains.com. Similarly, we obtained a list of known benign domains from Cisco Umbrella (top one million most popular domains).

Ultimately, we were capable of collecting approximately one million unique malicious domains, one million unique malicious IP addresses, and one million unique benign domains. Once we checked those against our event logs we had a total of 2.12 thousand matched malicious entities and 0.29 million matched benign entities.

Hardware Setup and Runtime Due to the fact that the entire modules are implemented in Apache Spark, it is possible to run the proposed approach on any configuration of hardware. In this regard, for the purpose of this research, we ran our experiment on a Spark cluster configured with 28-core 2.00 GHz GNU/Linux machine with 100 GB of RAM provided by the HPI Future SOC Lab (in shared access mode).

Using the dataset and hardware described above; the Extraction, Transformation, Ground Truth Construction, and Loading Modules all together take almost 50 minutes. In other words, 50 minutes to read the raw proxy and DNS logs as well as the collected Ground Truth (GT), preprocess them, and write a parquet file containing the prepared unique vertices and unique edges. Then the BP module takes almost 90 minutes to read that parquet file, construct the Markov network and run 7 iterations of the described BP.

4 Results and Discussion

Validation As mentioned before, one the GTC module's tasks is to carefully select a list of samples for the purpose of validation. In this regard, after constructing the Ground Truth which consists of a balanced number of matched malicious and benign entities (i.e. domains and IP addresses), the GTC module carefully marks n samples for *validation* and the rest for *training*. Then the BP module uses the training set to set up the priors and assigns an unknown prior to nodes marked for the validation.

This validation set must be chosen carefully as it is quite likely that a node presented in the validation set would have no path to any node presented in the training set therefore not allowing us to properly evaluate our system. This is due to the fact that it is quite rare to find two connected malicious entities both presented in a blacklist or a TI feed (e.g. finding a domain and its resolved IP address both listed in a retrievable list). The majority of these feeds, only list the malicious entities they have observed (e.g. the IP address of the phishing site, or the domain name for a malware hosting domain).

Hence, before taking the samples, we had to calculate the connected components in our GT and choose the validation samples from those. For example, if we take node i as a malicious node for validation, it must have a path to at least one other malicious node within the constructed graph. Furthermore, to hold a balance between benign and malicious nodes, we took half of the samples from malicious connected components and the other half from benign connected components.

We present our detection capability as Receiver Operating Characteristic (ROC) plot as shown in figure 3. This is achieved by thresholding malicious belief of nodes presented in the validation set. For instance, given a threshold t, and a node i, if i's malicious belief, $P(X_i = x_{mal}) > t$, then we predict i as malicious; else benign. This prediction is then compared to the i's original label to determine this detection as false positive, true positive, false negative, or true negative. After repeating this procedure for all the nodes in the validation set, it is possible to compute FPR and TPR for a given threshold t, and finally, plot the ROC based on different selections of t in the range of $[0, 1]$.

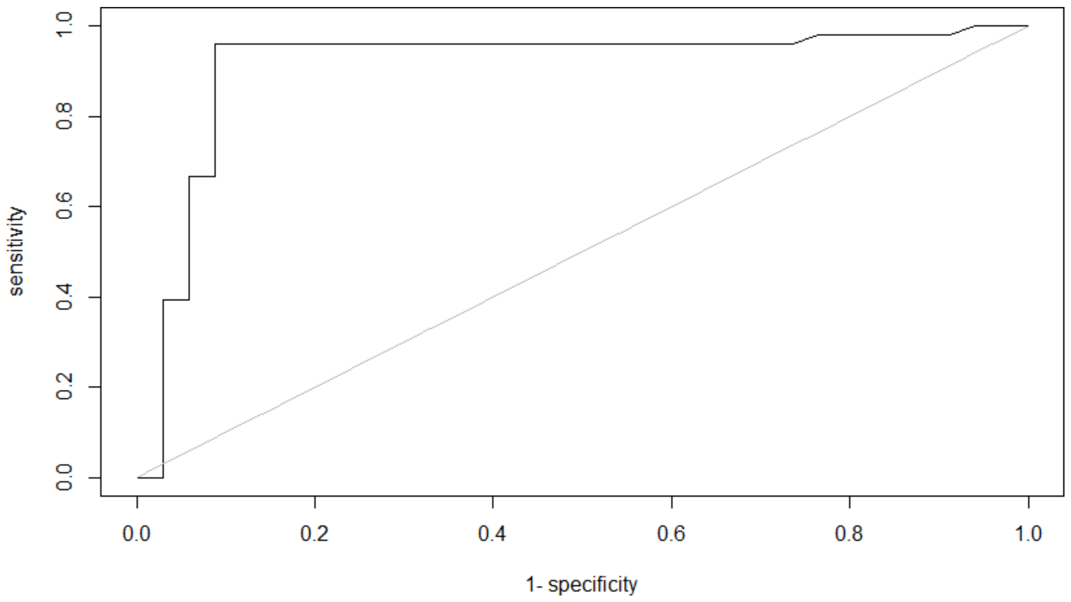


Figure 3: Receiver Operating Characteristics (ROC) Plot

As we can see the area under the ROC curve (AUC) is 91 % (the higher the AUC, the better the classification), while we achieve 96 % TPR with an 8 % FPR.

4.1 Analysis of False Positives and False Negatives:

To better understand the classification accuracy and the reason for the high FPR, we investigated the false positives (FPs) and the false negatives (FNs), i.e. benign domains that were wrongly classified as malicious and malicious entities that were wrongly classified as benign. We noticed the following observation when investigating these entities.

The majority of the FNs were entities associated with either cloud-based or online advertising services which introduce a significant challenge to our approach. For instance, we noticed that although some services were marked malicious in our ground truth, after running BP they were classified as benign due to their association with more benign entities. This decision making is reasonable as it does not fully make sense to mark an advertising platform completely malicious due to one malicious ad. Other FNs were malicious IP addresses that our system classified as benign, once investigated, we realized that although they may have been malicious at some point, that was no longer true. This classification can also be explained as these IP addresses were managed by cloud-based services that have reassigned those IP address.

FPs fell into two groups. First, entities that were classified wrongly due to a bad report from a TI source. For example, we noticed that there were some sub-domains that had their top-level domain wrongly blacklisted by Cisco Web Security Appliance[1] and therefore propagated to one of the threat intelligence sources we consumed, causing the subDomainOf relationship to overpower all the other *referredTo* relationships and eventually be classified as malicious. Second, entities that the system correctly identified as malicious despite the fact that they were labeled benign in GT. For instance, we were able to identify 6 entities that had turned malicious just recently and the system was capable of detecting those based the referredTo relationship to other malicious entities.

In summary, the FNs and FPs were mostly the result of bad GT (inaccurate threat intelligence), and the attempt of the system to correct that inaccuracy. This investigation shows that although it is better to validate the crawled and consumed threat intelligence, it is not crucial as such a system with more paths could potentially correct the TI inaccuracy.

Previously Unknown Malicious Entities We also investigated the ability of our approach to detect new malicious entities (i.e. entities that were not presented in the ground truth). After running the BP, we selected the top 100 entities that were assigned a high probability of belonging to class malicious and did not exist in the GT.

Although validating these entities is a challenging task (i.e. how one decides maliciousness), for the purpose of this research we manually validated those entities and concluded maliciousness if we observed a reputable threat intelligence source

[1] https://www.cisco.com/c/en/us/products/security/web-security-appliance/index.html (last accessed 2018-04-01).

(e.g. VirusTotal,[2] URLVoid,[3] AlienVault Open Threat Exchange[4]) reporting on that entity. It is also worth mentioning that for this task, we discarded the indicator's time stamp (i.e. the time in which that entity was seen). This is due to the fact that investigating and validating the maliciousness period is itself an extremely challenging task.

74 % of those entities were, in fact, malicious entities that did not exist in our GT, but have been reported as malicious by several other TI sources. These entities were mostly domains and IP addresses associated with services running on CloudFront.[5] 9 % were entities that we could not find any information about. They seem to be either services that were active for a short period of time and detected to be malicious based on their previous association with malicious entities, or the result of DGA. However, we could not validate those intuitions as there was no trace of them on the internet. The rest were entities that were classified wrongly.

In summary, investigating previously unknown malicious entities showed that our approach was capable of detecting new malicious entities that were not presented in our ground truth. Despite the fact that there were some misclassified entities (FPs), this approach is still extremely effective as blocking these FPs would not have a drastic effect, due to the fact that the main reason for classifying them as malicious was *not being associated* with major benign entities.

5 Future Work

One of the limitations of our work is the experimental setup. In this regard, we only had access to one day of proxy and DNS logs. It could be interesting to investigate how increasing the volume of the event logs would affect the detection. One could expect to have a better detection accuracy, as there would be more paths within the graph.

Furthermore, in this research, we only focused on three main relationships that directly connect domain names and IPv4 addresses. However, one could investigate indirect relationships such as *nameServerFor*, *mailServerFor*, *aliasFor*, with the intuition that cyber criminals tend to reuse infrastructure for their malicious entities. Additionally, it is also possible to take other host-based relationships into consideration (e.g. user agent, IP address, MAC address), thus enabling the propagation of the client machines' reputation to domains and IP addresses (a similar approach to [3, 8]).

In addition, it is also possible to look into enriched relationships such as registrar information, IP ranges, ASN, and BPG in order to construct a larger graph with a higher connectivity. Antonakakis et al. [1] make use of BGP, AS, and registration features as part of their feature set to detect malicious domains. [4] uses IP space

[2]https://www.virustotal.com (last accessed 2018-04-01).
[3]http://www.urlvoid.com (last accessed 2018-04-01).
[4]https://otx.alienvault.com/ (last accessed 2018-04-01).
[5]https://aws.amazon.com/cloudfront (last accessed 2018-04-01).

proximity to measure the similarity between domains. [6] and [2] investigate features such as name servers, and registrant information to detect malicious domains. In this regard, the investigation of these combined relationships pose an interesting direction for future work.

Finally, one could combine the global features (i.e. the features that would allow us to either directly or indirectly connect two entities) together with local features, such as, URL structure, port number, request/response length, etc. This combination of global and local features should, in theory, improve the accuracy.

6 Conclusion

In this research, we tackled the problem of detecting malicious domains and IP addresses by transforming it into a large-scale graph mining and inference problem. In this regard, we proposed an adaptation of belief propagation to infer maliciousness based on the concept of guilt-by-association using *subdomainOf*, *referredTo*, and *resolvedTo* relationships between IP and domain nodes. We evaluated our approach by running an adaptation of loopy belief propagation on a graph constructed from 2 TB of proxy and DNS logs collected from a global enterprise on hardware provided by the HPI Future SOC Lab. The results showed that our system attained a TPR of 96 % at 8 % FPR. While investigating the FP and FN we noticed the mistakes in the GT which was corrected by our system. We also investigated the system's ability to detect previously unknown malicious entities and demonstrated its capability to extend threat intelligence and blacklist by detecting new malicious entities.

References

[1] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. "Building a Dynamic Reputation System for DNS". In: *19th USENIX Security Symposium (USENIX Security 10)*. Washington, DC: USENIX Association, Aug. 2010. URL: https://www.usenix.org/conference/usenixsecurity10/building-dynamic-reputation-system-dns.

[2] M. Felegyhazi, C. Kreibich, and V. Paxson. "On the Potential of Proactive Domain Blacklisting". In: *Proceedings of the 3rd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*. LEET'10. San Jose, California: USENIX Association, 2010, page 6.

[3] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne. "Detecting malicious domains via graph inference". In: *European Symposium on Research in Computer Security*. 2014, pages 1–18. DOI: 10.1007/978-3-319-11203-9_1.

[4] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais. "Detection of early-stage enterprise infection by mining large-scale log data". In: *Dependable Systems and Networks (DSN)*. 2015, pages 45–56. DOI: 10.1109/DSN.2015.14.

[5] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. “All Your iFRAMEs Point to Us”. In: *17th USENIX Security Symposium (USENIX Security 08)*. San Jose, CA: USENIX Association, July 2008. URL: https://www.usenix.org/conference/17th-usenix-security-symposium/all-your-iframes-point-us.

[6] W. Xu, K. Sanders, and Y. Zhang. “We know it before you do: predicting malicious domains”. In: *Virus Bulletin Conference*. 2014, pages 73–77. URL: https://www.virusbulletin.com/virusbulletin/2015/02/paper-we-know-it-you-do-predicting-malicious-domains/.

[7] J. S. Yedidia, W. T. Freeman, and Y. Weiss. “Understanding Belief Propagation and Its Generalizations”. In: *Exploring Artificial Intelligence in the New Millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pages 239–269. ISBN: 1-55860-811-7.

[8] F. Zou, S. Zhang, W. Rao, and P. Yi. *Detecting malware based on dns graph mining*. International Journal of Distributed Sensor Networks, 2015. DOI: 10.1155/2015/102687.

City Sensing

Big Mobility and IoT Data Processing and Analytics in Urban Computing

Dragan Stojanovic, Aleksandra Stojnev, Igor Djordjevic, Natalija Stojanovic, and Bratislav Predic

Computer Science and Engineering Department
University of Nis, Faculty of Electronic Engineering
{firstname.lastname}@elfak.ni.ac.rs

With the widespread use of mobile devices with integrated sensors, massive amounts of data relevant to urban life and mobility related to traffic, air quality, and citizens' activities are collected and stored. In this report we present the CitySensing platform and several applications deployed at the HPI Future SOC Lab infrastructure to provide processing and analytics of Big crowd sensed and mobility data with the aim to monitor and detect situations and events that influence human mobility, health, and activities.

1 Introduction

The activities within the *CitySensing* project are focused on research and development of methods, tools, software systems and a platform for development of HPC and cloud-based applications that provide processing, analysis, and mining of big data collected by mobile crowd sensing and Internet of Things paradigms in Smart Cities.

With the proliferation of mobile devices with increasing computing, communication and sensing capabilities in everyday use, the important sources of sensor data become the users using such devices, bringing the ideas of citizens as sensors and mobile crowd sensing/sourcing to reality. Thanks to an increasing number of built-in sensors in mobile devices, user-generated geo-referenced content (video, photo, sound, text, etc.), and social networks/media interaction, citizens can collect massive amounts of data that, after processing, analysis, and mining could support numerous applications with the aim to make cities smarter. During the Future SOC Lab grant period we have explored the methods and technologies of IoT and mobile crowd sensing as Big mobility data sources, as well as appropriate methods and tools for efficient storage and management of massive amounts of such data in cluster/cloud infrastructure, i.e. Big mobility & IoT data. We have reviewed available methods, algorithms and tools for management, processing, analysis, and mining of Big crowd-sensed and mobility data on a cluster and cloud infrastructure, and decided to base our solutions on MapReduce/Hadoop [1], Apache Spark [3], and Apache Kafka [2].

After evaluation and collection of simulated and freely available Big data sets we have defined requirements for applications related to traffic control and remote health monitoring in a smart city scenario. We have developed, tested and evaluated several demo applications based on processing, analysis, and mining of big crowd sensed and mobility data in a Smart City related to traffic control, detection of popular places and remote health monitoring.

2 CitySensing

The *CitySensing* platform consists of mobile application components, server components and visualization/analytics components organized in a distributed architecture given in figure 1. It fully leverages processing, sensing and communication capabilities of mobile devices and provides distributed and scalable storage, processing, analysis, and mining of crowd sensed and mobility data at mobile devices within the *CitySensing* mobile components.

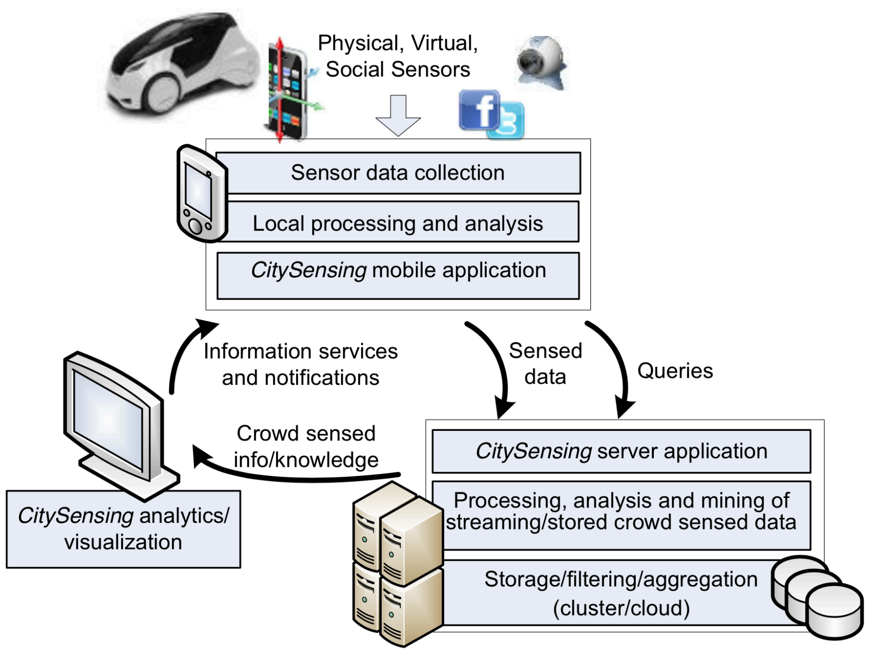

Physical, Virtual, Social Sensors
Sensor data collection
Local processing and analysis
CitySensing mobile application
Information services and notifications
Sensed data
Queries
Crowd sensed info/knowledge
CitySensing server application
Processing, analysis and mining of streaming/stored crowd sensed data
Storage/filtering/aggregation (cluster/cloud)
CitySensing analytics/ visualization

Figure 1: An architecture of the *CitySensing* platform

High-level mobility and sensor data generated at users' and IoT devices are fused, aggregated, processed and analyzed at the *CitySensing* server components running on a cluster/cloud infrastructure. For processing and analysis of Big crowd sensed and mobility data, the *CitySensing* server components are based on distributed processing frameworks, such as MapReduce/Hadoop and Apache Spark, for processing offline crowd-sensed data stored in a distributed file system over a cloud/cluster. For real-time processing and analysis of massive data streams, the Spark Streaming [3] data stream processing framework is employed. For communication between different *CitySensing* components Apache Kafka, a distributed message platform, is used.

2.1 Future SOC Lab infrastructure

For the purpose of deploying, testing and evaluation of *CitySensing*-based big data applications, we used following Future SOC Lab infrastructure:

- Cluster of 9 virtual machines: 1 master (XL VM), 8 slaves (L VM) installed and configured with Apache Hadoop, Spark and Kafka.
- 1000 Core cluster, using MPI on Slurm.
- GPU using NVidia/CUDA over Docker.

3 *TrafficSense*—a traffic monitoring application

The basic idea of *TrafficSense*, a traffic monitoring application built on top of the *CitySensing* platform, is to use drivers, their mobile phones, and their vehicles as moving sensor platforms that can collect traffic and road related conditions, events, and information and send them to a server/cloud infrastructure for processing and analysis. By using smartphone sensors, such as GPS, and open source road network and POI data (OpenStreetMap), the *TrafficSense* application can automatically detect the following traffic related events and conditions:

- Travel times over street segments
- Slowed down traffic (problem in a traffic flow)
- Traffic congestion(start-stop locations)
- Traffic stops

On the *TrafficSense* server(s), continuous measurement and monitoring of traffic conditions is performed. *TrafficSense* would also provide detection of aggregated mobility patterns and trajectories, collective activities and behaviour, as well as complex mobility situations: traffic congestions, risky and dangerous traffic events, frequent city routes, crowded evacuation paths in an emergency situation, popular places, and orders of visits, etc.

3.1 Big traffic data analytics

For realistic traffic data generation we use CrowdNav, a traffic simulation software based on SUMO [9] and TraCI (a Python interface to SUMO) that enables implementation and evaluation of big data self-adaptive systems simulation [4]. Docker images of SUMO/CrowdNav and Kafka are created and deployed at the master node (XL virtual machine) and run as Docker containers.

Within *TrafficSense*, we have implemented two applications that are deployed on the Future SOC Lab infrastructure. In the first one, we process and analyse offline traffic data generated by CrowdNav in order to detect an average travel times for street segments per minute in order to provide dynamic navigation with up-to-date travel costs.

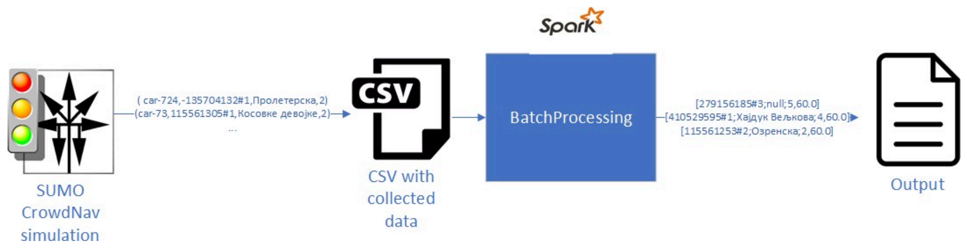

Spark
(car-724,-135704132#1,Пролетерска,2)
(car-73,115561305#1,Косовке девојке,2)
...
CSV
BatchProcessing
[279156185#3;null;5,60.0]
[410529595#1;Хајдук Вељкова;4,60.0]
[115561253#2;Озренска;2,60.0]
SUMO CrowdNav simulation
CSV with collected data
Output

Figure 2: TrafficSense application for calculation of travel times along the street segments

The second application is based on clustering of critical intersections in the city regarding the traffic congestions and provide online detection of traffic congestions, along with start and end times of the congestion. This application should provide rerouting of vehicles that approach the congested intersections and dynamic control of traffic lights and thus enables control of traffic flow and congestion around the city.

3.2 Calculations of dynamic travel times

To calculate average travel times for street segments for each minute, we have generated a large SUMO/CrowdNav dataset stored as a CSV file on a HDFS. The system architecture is shown in figure 2. The Spark job reads the records representing the vehicle's positions at particular time instants, detects the street segment the vehicle is on, and calculates the travel time for such a vehicle to pass a particular segment. By averaging travel times for each vehicle passing the particular segment, the application calculates average travel time for specific time period, a minute or longer, as specified in the application configuration. Such information is useful for providing dynamic navigation services.

3.3 Detection of traffic congestions

We use SUMO/CrowdNav simulator to simulate collection of real time vehicle positions, as in a system like Waze [10]. The vehicle/traffic data generated in real-time are published to particular topics of the Kafka message system, to which specific SparkStreamig jobs are subscribed. The application first calculates the centroid of the clusters, i.e. the street intersections, detected through the clustering process as potential traffic congestions locations. Then, through two separate processing/analytics pipelines and SparkStreaming jobs, the system detects the occurrence of traffic congestions in the real-time, as well as the start time and duration of each congestion. The system architecture is shown in figure 3.

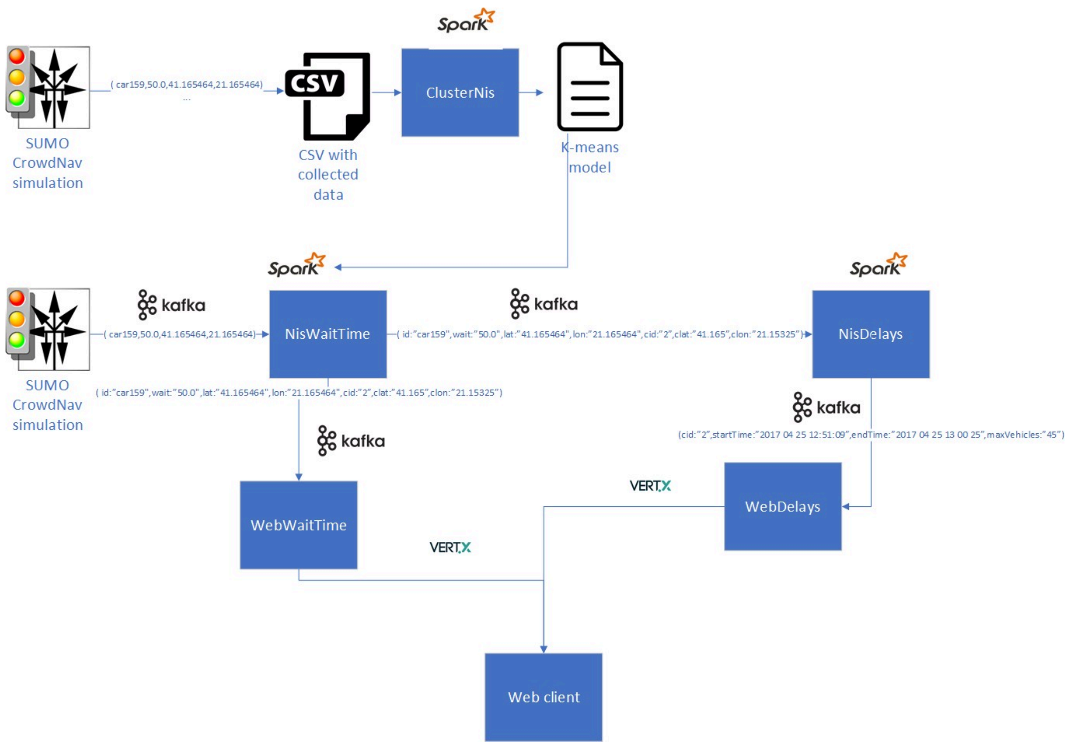

Spark
SUMO CrowdNav simulation
(car159,50.0,41.165464,21.165464)
...
CSV with collected data
ClusterNis
K-means model
Spark
Spark
kafka
kafka
SUMO CrowdNav simulation
(car159,50.0,41.165464,21.165464)
NisWaitTime
(id:"car159",wait:"50.0",lat:"41.165464",lon:"21.165464",cid:"2",clat:"41.165",clon:"21.15325")
NisDelays
(id:"car159",wait:"50.0",lat:"41.165464",lon:"21.165464",cid:"2",clat:"41.165",clon:"21.15325")
kafka
kafka
(cid:"2",startTime:"2017 04 25 12:51:09",endTime:"2017 04 25 13 00 25",maxVehicles:"45")
VERT.X
WebDelays
WebWaitTime
VERT.X
Web client

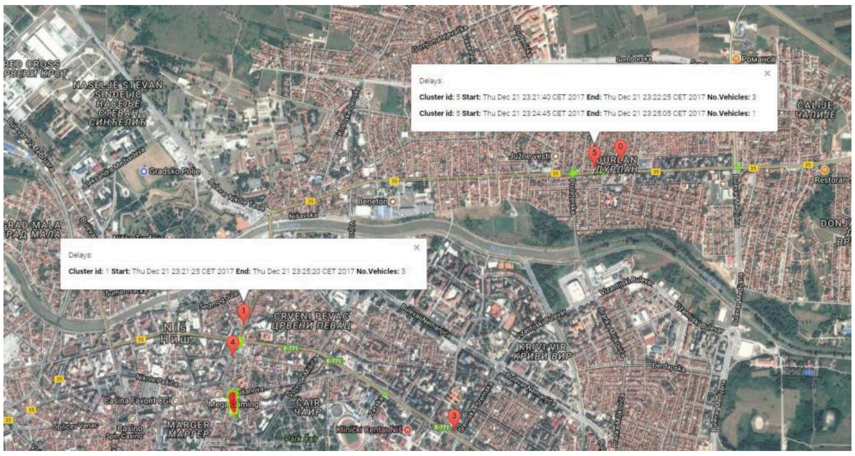
Figure 3: *TrafficSense* application for online detection of traffic congestions

3.4 *TrafficSense* visual analytics and dash-board

The CitySensing Web visualization/analytics components provide dashboard functionality through real-time monitoring of traffic and mobility of citizens in a Smart City intended for local authorities and traffic-related organisations. To provide dynamic visualisation of traffic congestions occurring in the city, along with start and end times of each congestion, we use Eclipse Vert.x, an event-driven application framework for JVM [6].

The Vert.x event bus allows web applications to communicate bi-directionally using WebSockets, providing development of real-time Web applications with server push functionality. We have implemented two Vert.x services: *WebWaitTime* and *WebDelays*, shown in figure 3.

The results of processing and analysis of SparkStreaming jobs within *TrafficSense* server application are visualized by the appropriate web application (figure 4). The demo dashboard shows parts of a Nis' street network with reported traffic conges-



Delays:

Cluster id: 5 **Start:** Thu Dec 21 23:21:40 CET 2017 **End:** Thu Dec 21 23:22:25 CET 2017 **No.Vehicles:** 3

Cluster id: 5 **Start:** Thu Dec 21 23:24:45 CET 2017 **End:** Thu Dec 21 23:25:05 CET 2017 **No.Vehicles:** 1

Delays:

Cluster id: 1 **Start:** Thu Dec 21 23:21:25 CET 2017 **End:** Thu Dec 21 23:25:20 CET 2017 **No.Vehicles:** 3

Figure 4: *TrafficSense* Web application

tions (heatmaps), their start and end times, as well as dynamic travel times over street segments.

We plan to collect a Twitter data stream in real-time with tweets located in the city area. We will try to perform correlation between traffic congestions and events on major city streets (shown by a heatmap) with bursts of 'traffic' on Twitter at the same locations. In this manner, the contents of tweets could offer a clearer and faster explanation of the causes of traffic congestions, e.g. a protest of local farmers.

3.5 GPU and 1000Core Cluster applications

To test and evaluate 1000 Core Cluster infrastructure and GPU playground with NVidia Tesla GPUs we have developed and tested two demo applications. The first one is developed using MPI library to work over 1000 Core Cluster machines through Slurm workload manager. The aim is to detect popular places in the city based on taxi pickup locations and times. The example data set is Uber trip data, which is obtained from the NYC Taxi & Limousine Commission.

Regarding GPU Lab infrastructure we are developing an NVidia CUDA application that uses the TensorFlow deep learning library to support image processing and object recognition. We have performed some test on handwritten documents but plan to extend our application processing and recognition of image data sets collected by drones (e.g. Stanford Drone Dataset [8]) in order to detect interesting objects and events from such images regarding urban mobility.

4 Ongoing work and next steps

Our ongoing work is related to remote healthcare and pervasive health concepts, and applications of largescale cloud-based systems for remote monitoring of people health and activities.

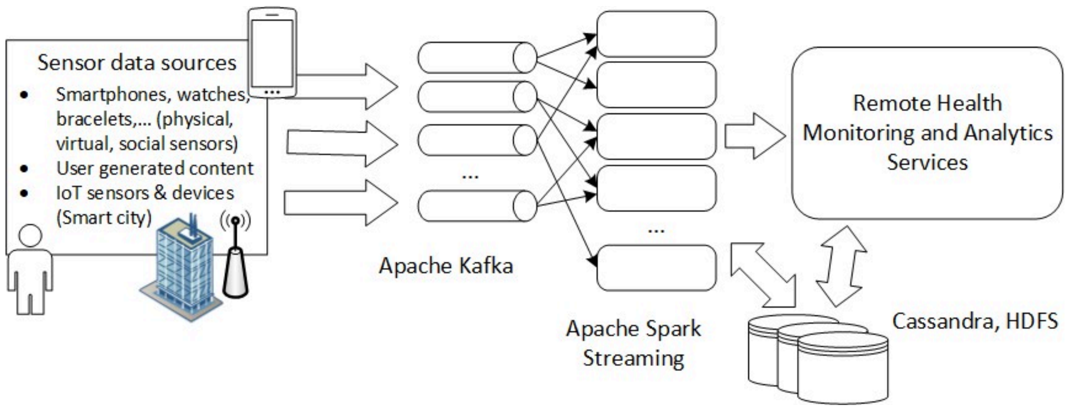

Sensor data sources
Smartphones, watches, bracelets,... (physical, virtual, social sensors)
User generated content
IoT sensors & devices (Smart city)
Apache Kafka
Apache Spark Streaming
Remote Health Monitoring and Analytics Services
Cassandra, HDFS

Figure 5: A general architecture of Remote Health Monitoring system

Taking into account all the challenges related to health/activities monitoring data and control flows that should be employed in a public, private or hybrid cloud, we are developing a Remote Health Monitoring architecture and implementing the system tailored to the analysis of health and activity data. The general architecture of Remote Health Monitoring system is given in figure 5.

We do not use a mobile health application for data collection, but we create a simulation of multiple users based on pre-recorded personal health/activities data, and use it for streaming such data to the system. Each user's simulation component reads a file that contains different records in csv format, and publishes them to the appropriate Kafka topic. Actual data are obtained from UbiqLog [7] and CrowdSignals datasets [5]. These datasets contain information retrieved from smartphones and smart bracelets, such as a heart rate, step count, accelerometer and gyroscope data, location, application usage and WiFi. A distinct record is generated for every measurement or event, containing user id, a topic, and a value. The id is used to separate different users, a topic to identify the type of measurement or event, and a value to hold actual data. For each type of record there is a predefined Kafka topic. These topic streams are monitored by a number of Apache Spark Streaming jobs. Each job acquires the data from monitored topics, and performs specific health- and activities-related analysis. We implemented a demo job for skin temperature prediction based on the heart rate and step count values. In offline mode, we first created a regression model for these parameters. Then, in a Spark Streaming job, we load that model and generate an alert in the form of a console notification if the predicted temperature is higher than the predefined threshold. This illustrates that the system can be used for complex event processing. Our vision is to create jobs that can detect complex events, extrapolate valuable information, and run spatio-temporal mining algorithms on available data. Furthermore, the visualization and alert-generating component of the system will be implemented.

5 Conclusions

We expect that availability of Future SOC Lab infrastructure for our project will further improve our research expertise and experience in domains of mobile crowd sensing, IoT and Big mobility and health-related data processing, analysis, and mining and will result in scientific achievements through preparation/publication of technical reports, scientific papers and development of prototype/demo solutions.

The availability of the Future SOC Lab infrastructure gives us useful insights in deployment and execution of big data applications on real cloud infrastructure.

References

[1] *Apache Hadoop*. URL: https://hadoop.apache.org/ (last accessed 2018-03-15).

[2] *Apache Kafka*. URL: https://kafka.apache.org/ (last accessed 2018-03-15).

[3] *Apache Spark*. URL: https://spark.apache.org/ (last accessed 2018-03-15).

[4] *CrowdNav - A model problem for big data self adaptive systems using SUMO and TraCI*. URL: https://github.com/Starofall/CrowdNav (last accessed 2018-03-15).

[5] *CrowdSignals.io - Building the Community's Largest Labeled Mobile and Sensor Dataset*. URL: http://crowdsignals.io/ (last accessed 2018-03-15).

[6] *Eclipse Vert.x - a tool-kit for building reactive applications on the JVM*. URL: http://vertx.io/ (last accessed 2018-03-15).

[7] R. Rawassizadeh and D. Kotz. "Datasets for Mobile, Wearable and IOT Research". In: *GetMobile: Mobile Computing and Communications* 20 (Apr. 2017), pages 5–7. DOI: 10.1145/3081016.3081018.

[8] *Stanford Drone Dataset*. URL: http://cvgl.stanford.edu/projects/uav_data/ (last accessed 2018-03-15).

[9] *SUMO – Simulation of Urban MObility*. URL: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-%209883/16931_read-41000/. (last accessed 2018-03-15).

[10] *Waze*. URL: https://www.waze.com/ (last accessed 2018-03-15).

Datalyzer
A web platform for streaming applications

Mario González-Jiménez and Juan de Lara

miso research group
Universidad Autónoma de Madrid (Spain)
Mario.GonzalezJ@uam.es, Juan.deLara@uam.es

Nowadays, streaming data are continuously generated from thousands of sources, including social networks, mobile apps, sensors, and many more. Hence, it becomes very useful to build applications able to process these data, with the purpose of filtering interesting parts, and monitor their runtime evolution. In this report we describe Datalyzer, an approach to create streaming data applications on the cloud based on a visual language. Datalyzer provides a facility to describe streaming data sources in an open way, and a visual language to describe the execution flow of the streaming application. In this report, we describe how the HPI SOC Lab infrastructure has helped us to develop and test Datalyzer and discuss oportunities for further developments.

1 Introduction

We live in a hyper-connected society, where 2.5 quintillions bytes of data are created every day.[1] Millions of heterogeneous sources around the world produce continuous streams of data, including sensors, social networks, IoT devices, mobile and web applications, among many others. Therefore, there is an increasing need to observe, monitor, analyse, detect anomalies [5] and make apps out of these heterogeneous data. However, currently, building such applications is only possible by engineers with specialized technical knowledge, and having expensive hardware at their disposal. Moreover, the required technologies for the demanding requirements of this kind of applications tend to have a steep learning curve and can be difficult to install, configure, and run [1].

In order to solve these issues, we are developing Datalyzer, a cloud-based platform to build streaming data applications. Our approach follows model-based development principles [2] and is based on a domain-specific visual language (DSL) to facilitate programming by non-experts. Datalyzer is able to synthesize code – based on Apache Kafka[2] – compile it, configure the application, and run it in an automated way. This way, the end user uses the web browser to visually create the model of the application to be built. As the approach is cloud-based, there is no need for difficult

[1] https://www.domo.com/learn/data-never-sleeps-5 (last accessed 2018-04-01).
[2] https://kafka.apache.org/ (last accessed 2018-04-01).

configurations by the end user. Moreover, the approach is extensible, and relies on a common format to describe live data sources. This permits the addition of new heterogeneous sources as needed, which can then be used in Datalyzer programs.

In this report, we describe the main principles of Datalyzer, findings about the application using HPI Future SOC LAB resources and next steps proposed for our project. More information on Datalyzer can be found at [4].

2 Datalyzer

This section describes the main ingredients of Datalyzer: a uniform, extensible representation of streaming data sources (section 2.1) and a visual DSL to describe execution workflows (section 2.2).

2.1 Data Sources

Dynamic data sources available on the web are heterogeneous with respect to the protocols and technologies used, and their access architectures, typically either pull-based (polling) or push-based (streaming) [3]. Our goal is to be able to create dynamic data streams out of any source type, and hence our approach permits abstracting away the technical details of the different sources (e.g. whether pull or push). This way, we can use them as dynamic live data channels in a uniform way. For this purpose, we have desinged a declarative description model for data sources, which facilitates the incorporation of new sources. A scheme of the description model is shown in figure 1. DataSourceType describes different types of data transmission technologies like TCP Socket or REST, but also non-standard technologies like Satori [3] or Twitter.[4] Once a DataSourceType (e.g. REST) is described, it is possible to describe DataSources of such type (e.g. OpenWeatherMap).

2.2 DSL

Once the data sources of interest are defined, the user can create her application using a visual DSL we have designed. In practice, this creation process occurs within a web browser, using a drag and drop mechanism. After finishing the description of the application, a code generator we have built synthesizes the program code, compiles it, and finally executes the described streaming application. By using the DSL, the user can instantiate data sources among those previously defined.

For example, if she defined a TCP socket data source (data source type: socket) and a OpenWeatherMap[5] data source (data source type: REST), she can instantiate and use them in the application she is creating. When a data source is added to

[3]https://www.satori.com/ (last accessed 2018-04-01).
[4]https://developer.twitter.com/ (last accessed 2018-04-01).
[5]https://openweathermap.org/api (last accessed 2018-04-01).

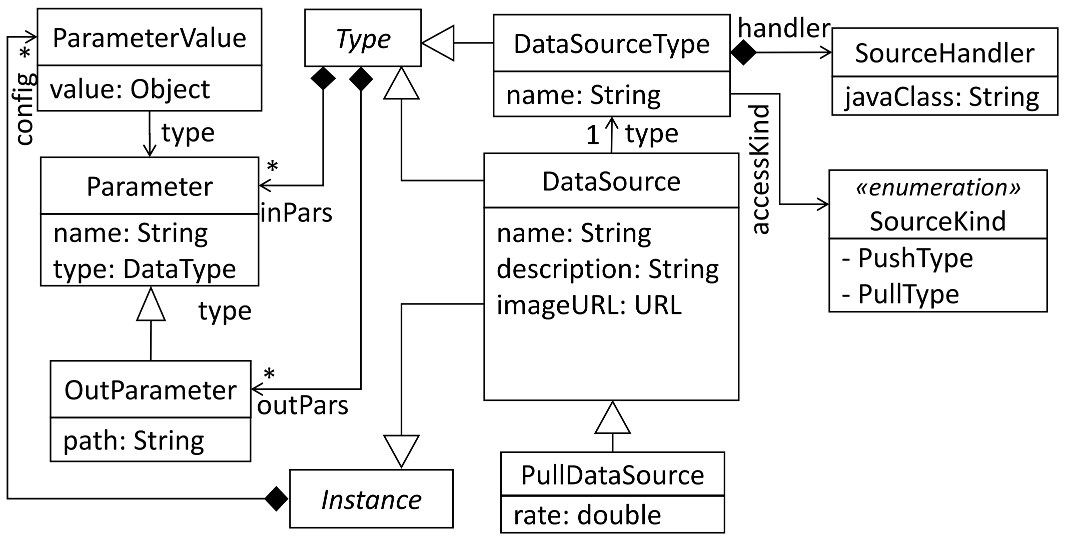


Figure 1: Conceptual model (excerpt) for dynamic data sources

the application, the user needs to configure input parameters defined in the model (figure 1) as a specific port for a TCP socket or a city name for OpenWeatherMap. Each data source must be connected with a dataflow pipeline where the data will be received. Over a pipeline, the user can apply transformations such as filtering, selecting values of interest or joining data from different data sources. A pipeline can be connected to other pipelines, but must be connected to at least one terminal node which produces a visualization for the data, a trigger, or persists the data. There are different types of visualizations for data that a user can select, such as tables or dynamic bar and line charts. In the future, we want to implement an output streaming data channel in order to make it possible to use Datalyzer as a service in real-time for external applications. An example of a simple application created with DSL is shown in figure 2. When the user has described her application including data sources, pipelines and terminal nodes, she can compile it and run it by accessing to the dashboard panel which is explained in the next section.

3 Architecture

Datalyzer is composed of several components and technologies. In order to understand our goal when using the HPI FutureSOC Lab infrastructure, the difficulties of the project and complexity of deploying it on the cloud, this section gives an overview of the architecture of Datalyzer that is shown in figure 3. The components are as follows:

Web Platform A web platform developed using NodeJS and MongoDB (see label 1 in figure 3). In this platform, users may create projects that contain the models

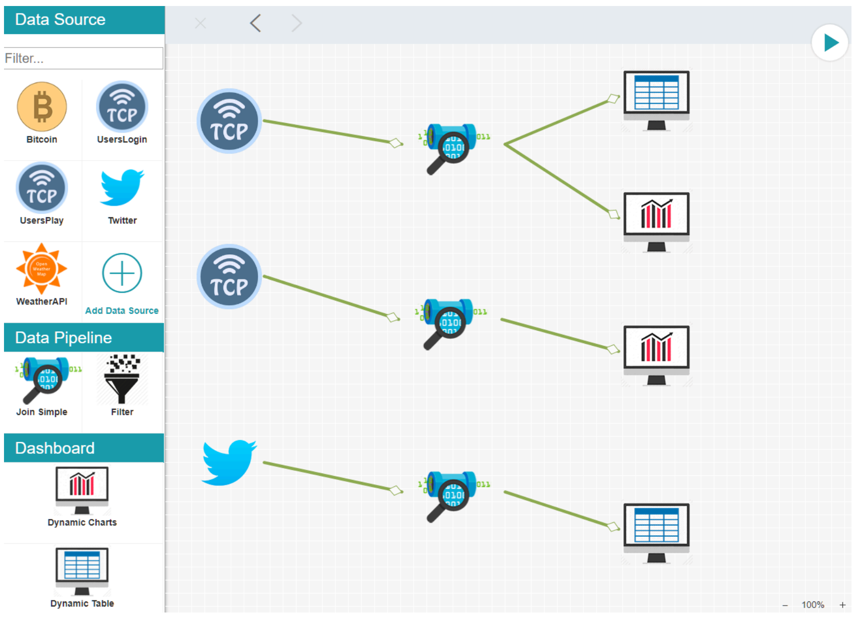


Figure 2: An example application created in the DSL with three data sources, three pipelines and several chart visualizations

of the streaming applications. The platform permits managing these projects, listing them, editing and executing the models.

Data Source Repository The description of data source types and data sources (seen in section 2.1) are stored in a MongoDB database (label 2). This database is connected with the DSL in order to show the different data sources in the editor palette.

DSL The DSL editor is integrated in the web platform (label 3). This editor is accessed whenever a project is edited. The DSL editor is built in Javascript and contains built-in validations to ensure (syntactical) model correctness at design time. If a model is correct, it can be stored in the database so that it can be opened, modified or executed later.

Code Generation We have built a code generator in Java that is invoked by the web server whenever the application is to be executed (label 4). The end result is an independent, fully functional Java project (based on Apache Kafka), which expects to receive and process the live data sources as specified (label 5).

Streaming Application The generated application (label 6) is executed on the sever. It uses Apache Kafka as a basis to create the data channels. This technology offers fast data pipelines, which are fault tolerant, scalable and distributed. The generated code processes these data channels as specified using the DSL. Once a data channel is initialized and opened, it waits to receive information from the data sources. When data is received, these are inserted in the data channel and processed. This continues until the application is paused or stopped.

Dashboard The streaming applications are executed in a server, so that the user does not have direct access to them. To allow interaction, we have developed a graphical, browser-based dashboard, which is integrated in the web platform (see label 7). The developed interface contains controls to monitor the application status (running on/off); management controls to execute, pause and stop the application; and several widgets developed in JavaScript to visualize the data (e.g. table, dynamic chart).

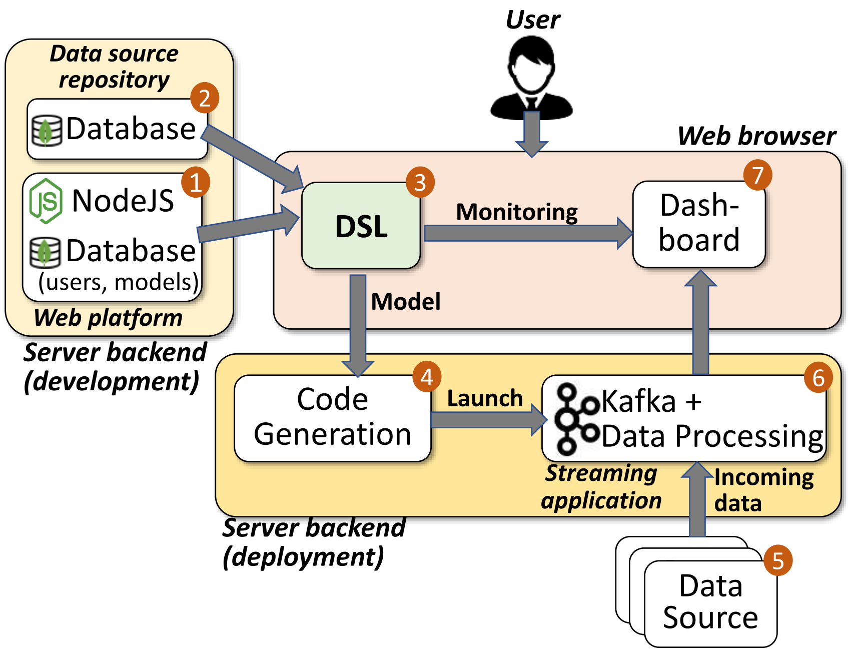


Figure 3: DATALYZER architecture

4 Use of HPI Future SOC Lab resources

We have been assigned 3 virtual machines with Ubuntu 16.04, 8 GB, 4 x Intel Pentium 4 @2.4 GHz cores and 100 GB HDD. The main purpose of these resources is to prepare a pre-production enviroment in the cloud for building an appropriate test environment for our project. We have designed several cloud tests in order to check performance, stability of the system developed, and to detect possible bottlenecks in any component (see previous section) that can imply a redesign of the architecture. The development environment in our group lacks access to a cloud infrastructure and this testing process is crucial to know the technical limits of DATALYZER. Furthermore, by using three virtual machines, we also can check the application in a distributed system. For example, NodeJS allows several strategies of scalability (e.g. splitting tasks in several instaces or using the cluster module), which must be evaluated.

4.1 Findings

Unfortunately, our project has been delayed with respect to the project schedul, due to the heavy development efforts needed to develop DATALYZER. Althought we have developed a completely functional first beta version of DATALYZER, a full deployment on our assigned resources has not been possible at this time. The web platform has been deployed and evaluated with positive results, including the integration with the DSL editor, database connection and code generation. However, the integration with the streaming application has not yet been reached.

Although we work with multi-platform technologies, some features are launched in an autonomous way using scripting. Commands and the way to invoke them are not the same in Linux (pre-production environment) and Windows (development environment). As we want to develop a multi-platform application, this problem has to be resolved. We are considering the creation of configuration files so engineers can manually configure them with regards the environment used, but this solution is not optimal. Another idea would be to build a container in Docker[6] which offers portability across cloud environments by packaging the application with all of the components and dependencies it needs.

4.2 Next steps

Our next step is to make DATALYZER multiplatform to be able to achieve a successful deployment in Linux servers. As we did not have the time to perform the experiments we designed, we have applied for an extension of our project. Now that we have a stable, fully functional version of DATALYZER, performing these experiments should be doable in the envisioned time frame.

5 Conclusions

In this report, we have described DATALYZER, a cloud-based approach for the creation of streaming data applications, based on a visual DSL. The approach permits a visual description of the application, which is then compiled, deployed, and executed automatically. The approach is extensible, as new data sources can be added on the fly. Overall, the approach aims at lowering the entry barrier for the creation and execution of streaming data applications.

We have partially deployed DATALYZER on the infrastructure facilitated by the HPI FutureSOC Lab, but further testing of the performance of the generated applications would be desirable. For this reason, we have applied for an extension of the project.

[6] https://www.docker.com/ (last accessed 2018-04-01).

References

[1] M. D. de Assunção, A. D. S. Veith, and R. Buyya. "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions". In: *J. Network and Computer Applications* 103 (2018), pages 1–17.

[2] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice, Second Edition*. Synthesis Lectures on Sw. Eng. Morgan & Claypool Publishers, 2017.

[3] A. Harth, C. A. Knoblock, S. Stadtmüller, R. Studer, and P. A. Szekely. "On-the-fly Integration of Static and Dynamic Sources". In: *COLD*. Volume 1034. CEUR Workshop Procs. 2013.

[4] M. G. Jiménez and J. de Lara. "Datalyzer: streaming data applications made easy". In: *ICWE*. 2018, in press.

[5] L. Rettig, M. Khayati, P. Cudré-Mauroux, and M. Piórkowski. "Online anomaly detection over Big Data streams". In: *2015 IEEE Int. Conf. on Big Data*. IEEE, 2015, pages 1113–1122.

AutoDVT

Real-time Classification for Vein Compressibility Analysis in Deep Vein Thrombosis Ultrasound Diagnostics

Sven Mischkewitz and Bernhard Kainz

ThinkSono
sven@thinksono.com

1 Introduction

Deep vein thrombosis (DVT) is caused by the formation of a blood clot within a deep vein, that most commonly takes place in the leg. If left untreated, DVT may lead to serious complications, including pulmonary embolism, which develops when pieces of blood clot break loose into the bloodstream and block vessels in the lungs. Typically, an average of one in a thousand people will be affected by DVT and related conditions during their lifetime. 20 % of patients die because of DVT-related complications. Worldwide, approximately 10 million people suffer from DVT or related conditions, estimates suggest that 100,000 Americans alone die of DVT each year [1]. Patients are referred for DVT-specific tests by front line medical professionals or following surgery. As the risk of DVT leading to serious complications, including death, is high, development of DVT-focused point-of-care diagnostics is of great importance.

There are two major challenges in diagnosing DVT. First, DVT does not necessarily show evident symptoms and the symptoms may overlap with other, less serious conditions, making it impossible to discover DVT without clinical tests. Second, the clinical routine method used to diagnose DVT is a D-dimer blood test [2], which determines the concentration of a small protein fragment in the blood that occurs when a blood clot is degraded by fibrinolysis. The D-dimer blood test can with high certainty rule out a pulmonary embolism (PE), the main DVT-related complication leading to death. However, while this test shows a high sensitivity to PE, it has a low specificity to early DVT and returns a high number of false positives. Patients with false-positive D-dimer results are referred to unnecessary further expert examinations, which is costly and time consuming. Furthermore, the consequently introduced workload is very large relative to the number of specialist DVT radiologists.

A more accurate screening method is the manual evaluation of vein compressibility during ultrasound (US) examination of standardized anatomical locations (usually at three specified landmarks on the femoral and popliteal veins [3], C-US method). However, for front line medical professionals, it is currently difficult to assess DVT using US since training is required to navigate to anatomically defined landmarks on the veins where compressibility has to be evaluated.

Contribution To solve this problem, we propose an automatic, point-of-care ultrasound (POCUS) image-based method to make DVT diagnostics accessible for front-line non-specialists. We propose a dual-task convolutional neural network (CNN) that jointly classifies the anatomical landmark plane in the current field-of-view and scores vein compressibility. Thus, the proposed AutoDVT network can intrinsically learn to interpret video data to perform localisation, segmentation, local deformation estimation, and classification from weak global labels. Furthermore, it is designed to require few floating point operations to enable real-time performance, and its performance is thoroughly evaluated on over 100 real, manually-annotated ultrasound sequences from DVT examinations.

2 Data collection

We have collected US data from 115 healthy volunteers using a Clarius L7 Handheld Wireless Ultrasound Scanner, a Phillips iU22, a GE Logiq E9, and a Toshiba Aplio 500. The dataset consists of 1150 sequences with each sequence having 100 to 200 frames. The frames have been labeled by 25 skilled annotators (medical students) and reviewed by a specialised radiologist. Labelling of each frame consists of a) defining the landmark location and b) manual segmentation, deciding whether each pixel of the frame corresponds to vein, artery or background. In this work we train a model to predict two tasks:

- The landmark location. In particular we focus on identifying 2 landmark locations, Saphenofemoral junction (LM1) and great saphenous vein (LM2), versus background frames defined as frames that do not contain a landmark.
- Vein compression. We identify whether the vein is open or closed. The vein in the training data is defined to be open or closed based on the number of vein pixels present in each frame from the manual segmentation.

We split the dataset using 80 % of the sequences for model training and the remaining 20 % as test data.

3 Methods

The AutoDVT tool utilises a CNN model for the prediction of landmark location and vein compression. In the following sections we present the individual components of the framework with respect to the data processing, the model design and training process.

3.1 Data Processing

3.1.1 Data Design

To take into account the noisy nature of the US, we train the model based on a number of consecutive frames instead of a single frame. This allows to incorporate temporal information in the network and can alleviate problems in single frames due to noise and artifacts, which are ubiquitous issues with heated US probes.

3.1.2 Data Augmentation

Data augmentation is used to increase the effective size of the training data using different transformations. It allows the model to be invariant to these transformations and allows to generalise to different input data distributions. We augment our data by applying a random affine tranfsormation to each block of consecutive frames each time they are used for training. In particular, we randomly apply one or more of the following transformations: horizontal flipping, translation, rotation, scaling.

3.1.3 Data Sampling

The different labels in the training data (Open, Closed, LM1, LM2, BG) have different sample frequencies. For instance, image frames belonging to close state class are significantly outnumbered by the open state class. This imbalanced distribution is problematic as the model will be biased towards the most frequent classes. In order to handle this problem during training, we sample each class equally.

3.2 Classification Model

3.2.1 Model Design

The classification model is a fully convolutional neural network defined by a series of convolutions. Since the network input is a number of consecutive frames stacked together, we initially use a number of 3D convolutions to fuse the temporal information (3×3×3 convolutions). In the experiments presented here we use 9 temporal frames and therefore we use 4 3D convolutions after which the temporal dimension is reduced to 1. Afterwards we perform 2D convolutions until the feature size is reduced to 2×2 (3×3 convolutions). All convolution layers (both 3D and 2D) are followed by rectified linear (ReLU) non-linearity and batch-normalisation to enhance the convergence and accuracy of the network. Finally we utilise a global average pooling (GAP) layer and a fully-connected (FC) layer to learn the final class probabilities.

The input is downsampled with each 3D convolution to allow to learn features at different resolutions and reduce the computational cost of the model. This is done using strided convolutions for all the 3D convolutions. We propose a combined model that learns to predict both the landmark location and the vein compression. To achieve this we use a common trunk of convolutions for both tasks (4 3D convolutions and 1 2D convolution) and then use a separate branch for each task (2 2D convolutions, 1 GAP and 1 1 FC layer). Using this design we force the two tasks to share common low-level features through the common trunk, while allowing to learn task-specific features through each separate branch. Additionaly, the time required to compute the labels for the two tasks is reduced by half by using a single model instead of two separate models. Drop-out (50 % rate) is used after the convolutions at the task-specific branches to improve the regularisation of the network.

3.3 Model Optimisation

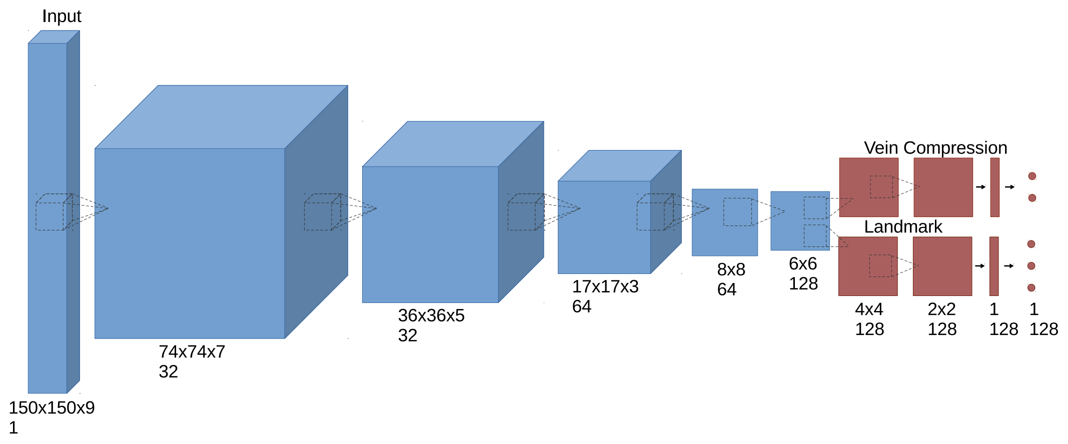


Figure 1: Architecture of the proposed model. The common trunk of the network is depicted in blue and the individual task branches in red. Under each layer we specify the layer dimensions and the number of filters.

The classification model is trained using the standard ADAM optimiser (initial learning rate of 10^{-3} and $\beta = [0.9, 0.999]$). The ADAM optimiser utilises a momentum term to scale the gradient updates and has been shown to converge to minima faster than standard SGD optimisers, without having to manually adjust the learning rate. The model is trained using the sum of the cross-entropy from the vein compression classification and landmark detection tasks. Additionally, we use L2 norm regularisation on the weights (coefficient 10^{-5}) to avoid model over-fitting.

Since the two different tasks have a different number of samples, we perform sampling in two stages. Initially, we randomly select either of the two tasks with equal probability. Then we draw a sample based on the different labels of the task with equal probability for all the task labels. The network is trained for 50 epochs with 12 000 samples for each epoch.

4 Evaluation

We evaluate the performance of AutoDVT on the test data based on the F1 score, the harmonic mean of precision and recall. The F1 score is computed independently for the two tasks. The model achieves average F1 scores of 88 % for the vein compression

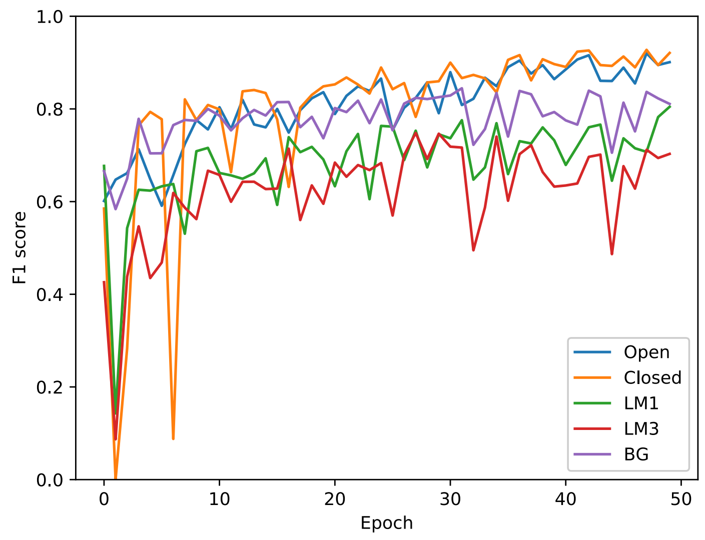


Figure 2: F1 scores on a sample of the test data (1500 samples) during training.

task (open: 90 %, closed: 85 %) and 78 % for the landmark detection task (LM1: 75 %, LM2: 65 %, BG 0.93 %). Figure 2 shows the F1 score computed on a subset of the test data during training.

The model achieves real-time performance with 20 frames-per-second (fps) on a single GPU and 14 frames-per-second on a 4-core CPU (a K80 GPU and Xeon E5-2686v4 CPU was used in our experiments). In practice, the frame rate is limited by the image acquisition rate of the ultrasound probe, which is around 20 fps.

5 Conclusion

We have proposed a model that identifies the landmark location and the vein compression from US frames. The model achieves real-time performance even on CPU and achieves accuracies of around 90 % and 80 % for the different tasks. Using the predictions from the network we are now designing the front-end of the system that will allow guidance of non-expert health practitioners to reliably diagnose DVT directly at the point-of-care.

In parallel, we are collecting additional data and further optimising the network architecture and parameters to improve even more the accuracy of the network. Finally, we aim to generalise the network to identify all relevant landmarks along the femoral vein.

References

[1] M. G. Beckman, W. C. Hooper, S. E. Critchley, and T. L. Ortel. "Venous Thromboembolism: A Public Health Concern". In: *American Journal of Preventive Medicine* 38.4, Supplement (2010). Blood Disorders in Public Health, S495–S501. ISSN: 0749-3797. DOI: https://doi.org/10.1016/j.amepre.2009.12.017.

[2] "d-Dimer for the Exclusion of Acute Venous Thrombosis and Pulmonary Embolism". In: *Annals of Internal Medicine* 140.8 (2004), pages 589–602. DOI: 10.7326/0003-4819-140-8-200404200-00005. PMID: 15096330.

[3] S. M. Schellong, T. Schwarz, K. Halbritter, J. Beyer, G. Siegert, W. Oettler, B. Schmidt, and H. E. Schroeder. "Complete compression ultrasonography of the leg veins as a single test for the diagnosis of deep vein thrombosis". In: *Thromb Haemost* 89.02 (2003), pages 228–234. ISSN: 0340-6245. DOI: 10.1055/s-0037-1613436.

Continuous Performance Testing for Microservices

André van Hoorn, Vincenzo Ferme, and Henning Schulz

University of Stuttgart
Institute of Software Technology

This report provides a summary of our project "Continuous Performance Testing for Microservices" conducted during the HPI Future SOC Lab period fall 2017, as well as ideas for a follow-up project for the upcoming period.

1 Introduction

Modern software engineering paradigms and technologies—such as DevOps [3] (including automation as part of continuous delivery) and microservices [9]—are gaining more and more attraction in the software and services engineering communities. Of particular interest are quality-of-service concerns, for instance, w. r. t. performance and reliability. While established approaches for classic contexts (i.e. which do not use DevOps and microservices) exist, their adoption to DevOps and microservices requires considerable research efforts [4, 7].

In the recent years, our group has already contributed architecture-aware approaches for performance and reliability, involving a combination of measurement-based and model-based techniques [8, 10, 12]. Recently, we started to investigate how these techniques can be used in DevOps and microservice contexts—with a particular focus on load testing as a key performance engineering activity [5, 11, 12].

In order to conduct large-scale experimental evaluations, we need a state-of-the-art computing infrastructure such as the one provided by the HPI Future SOC Lab.

We have requested the HPI Future SOC Lab for setting up an infrastructure for the experimental evaluation of our ongoing research on continuous performance testing for microservices. We have adopted the BenchFlow [6] benchmarking framework to support microservices and used it for extensive exploratory performance testing of microservices based on a novel approach.

In the remainder of this report, we will list the granted Future SOC Lab resources, provide a brief description of our conducted activities as part of the project, and outline next steps.

2 Granted Future SOC Lab Resources

We requested and received dedicated (root) access to the following computing resources (servers): *i.*) 896 GB RAM, 80 cores; *ii.*) 32 GB RAM, 24 cores. Dedicated access has been given to us due to our expected high resource demands.

3 Project

The project goals as stated in the proposal were threefold:

Using BenchFlow for Declarative Load Testing of Microservices We will continue our efforts to use BenchFlow for load testing microservices. We will use one or more sample microservice applications, such as the Sock Shop. During the SOC lab period Spring 2017, we have already started to setup the infrastructure and will be able to continue from this point on. We plan to integrate BenchFlow also into our attempts on Declarative Performance Engineering [13].

Automatic Extraction of BenchFlow Load Test Specifications from APM Data In our previous work, we have developed the WESSBAS approach [12] for specifying, extracting, and generating workload specifications for application systems. We will extend the workload extraction from APM data [6] to microservices applications and develop a transformation to BenchFlow load test specifications. The experimental evaluation will be concerned with the accuracy of the extraction and load generation.

Efficient Performance Testing of Microservices Using Markov Chains It is not feasible to execute the complete set of available performance tests for each software change as part of the continuous integration automation. Hence, we are interested in selecting and executing only relevant tests. We plan to adopt a seminal approach for efficient selection of performance tests from the telecommunication domain [2] for DevOps and microservices. Particularly, we will use the Markov chain based approach to identify and execute selected load tests using our previously described WESSBAS/BenchFlow approach.

4 Description of Activities

4.1 Sock Shop Microservices Application

As a system under test, we have used the Sock Shop developed by Weaveworks.[1] It represents an e-commerce website and is available under an open-source license. The distributed application has been designed and implemented based on the microservices architectural style [9] and respective state-of-the-art technologies. Each microservice is available as a Docker image, which is a container-based virtualization technology. A recent survey identified the Sock Shop as a candidate for a benchmark application for microservices [1], which was a main reason for choosing it.

We have used the application as the system under test for the following three projects:

[1]https://microservices-demo.github.io/ (last accessed 2018-04-01).

Using BenchFlow for Load Testing of Microservices We started to use BenchFlow as an execution framework for automating the load test execution of services deployed in Docker containers on top of the Rancher[2] orchestration framework, by relying on the provided Declarative Domain Specific Language for defining performance tests. We have executed extensive experimentation to assess BenchFlow's capability in providing reliable and reproducible results on the HPI infrastructure.

Automatic Extraction and Evolution of Load Test Specifications from APM Data Building on our WESSBAS approach [12], we worked on automating the load testing process. Instead of specifying a load script that is executed, we utilize APM data to generate a load test representing the real users' behavior. However, manual adjustments are often required to specify input data or implement ID correlations, hindering automation. For this reason, we developed an approach to store manual adjustments to generated load tests separately which allows to generate currently representative load tests on the fly (e. g. in a continuous integration pipeline) without manual intervention [11].

Efficient Performance Testing of Microservices Using Markov Chains We executed extensive exploratory performance testing to collect performance data to be used to assess the feasibility of using Markov chains to efficient performance testing of microservices. We explored multiple dimensions on the possible performance space of the system under test, as for example impact of varying the amount of resources (RAM, CPU) available to the different services, configurations of those services, and the number of replicas of each service.

5 Next Steps

In the next period, we want to continue this line of research and experimentation. We want to focus particularly on the aspect of *DevOps-oriented Declarative Load Testing for Microservices* and plan the following works:

5.1 Declarative Load Testing

We started to use BenchFlow as an execution framework for automating the load test execution. We have executed extensive experimentation to assess BenchFlow's capability in providing reliable and reproducible results on the HPI infrastructure. We plan to integrate BenchFlow also into our attempts on Declarative Performance Engineering [13]. For doing so we are going to extend the declarative language and execution framework provided by BenchFlow, and relying on extensive experimentation to assess the possibility to abstract complexity from the developers and performance engineers when executing complex performance test activities.

[2]https://rancher.com (last accessed 2018-04-01).

5.2 Detection of Performance Regressions based on Load Tests

One use case of load testing is the detection of performance regressions that have been introduced during the development. We plan to develop a declarative approach for microservices, so that developers can specify the regression criteria to be met and detect violation of those during continuous development of software systems. To accomplish so, we plan to rely on state-of-the-art techniques for regression detection and execute extensive experiments to assess the capability of the declarative approach to correctly and automatically report regression issues. As part of this, we plan to experimentally assess the suitability of performance metrics for regression detection of microservices.

5.3 Prioritization and Selection of Load Tests

Load tests are resource-intensive and time-consuming. Hence, it is infeasible to execute the entire test suite for each software change as part of the continuous delivery pipeline. Instead, it is desirable to run only those tests that are (performance-)relevant for the most recent change. We plan to continue our activities for efficient load testing started in the previous period, and investigate additional techniques for prioritizing and selecting load tests. For example, we plan to mine the data collected by BenchFlow about past executions, and perform additional experiments by deploying the services on top of a Kubernetes cluster, with the aim of defining a Key Performance Indicator (e. g. maximum duration of tests, maximum use of resources, or desired test coverage) enabling us to prioritize the execution of performance tests, when no time is available to execute the entire test suite. An additional input to the algorithms is contextual data, e. g. information about the usage profile expected during deployment of the next version. We then plan to provide a declarative specification to perform this kind of performance analysis, and integrate the possibility to automate the mentioned analysis by relying on the BenchFlow framework.

5.4 Advanced Extraction of Load Test Specifications from APM Data

Specification of a load test that is representative for the actual user behavior is challenging. In our previous work, we developed an approach to generate representative load test specifications from APM data using the WESSBAS approach [12]. As an advancement of this, we plan to take special account of the evolution of the monitored system as well as of microservice architectures. Since the behavior of the users of the target system as well as the system interface can change, we want to semi-automatically detect and merge these changes into already existing load tests. Furthermore, we are going to modularize load test specifications to apply the load test generation approach to single microservices or subsets of all of the target systems microservices.

5.5 Case Studies

As a system under test, we have so far mainly used the Sock Shop developed by Weaveworks. It represents an e-commerce website and is available under an open-source license. In order to increase the generalizability of our results, we plan to experiment with additional microservices-based systems under test. Possible candidates are the Pet Supply Store[3] or the next version of the SPECjEnterprise® industry-standard benchmark developed by the Standard Performance Evaluation Corporation.[4]

6 Conclusion

We are thankful to the HPI Future SOC Lab for having granted us access to the computing infrastructure. The environment eases the joint work of different organizations on the platform, which has so far been hindered by university-internal access constraints — apart from the fact that an equipment comparable to that of the HPI Future SOC Lab has not been available to us, and that enables extensive performance configuration tests needed to reach our goals.

7 Acknowledgment

The presented works are a joint effort being conducted together with many researchers from different organizations, including (in alphabetical order) Alberto Avritzer, Andrea Janes, and Barbara Russo.

References

[1] C. M. Aderaldo, N. C. Mendonc¸a, C. Pahl, and P. Jamshidi. "Benchmark requirements for microservices architecture research". In: *Proceedings of the International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE 2017)*. 2017, pages 8–13. DOI: 10.1109/ECASE.2017.4.

[2] A. Avritzer and E. J. Weyuker. *The automatic generation of load test suites and the assessment of the resulting software*. Eng., 21(9): IEEE Trans. Softw, 1995. DOI: 10.1109/32.464549.

[3] L. J. Bass, I. M. Weber, and L. Zhu. *DevOps — A Software Architect's Perspective*. Addison-Wesley, 2015.

[3] https://github.com/DescartesResearch/Pet-Supply-Store (last accessed 2018-04-01).
[4] https://www.spec.org/ (last accessed 2018-04-01).

[4] A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. Herbst, P. Jamshidi, R. Jung, J. von Kistowski, A. Koziolek, J. Kroß, S. Spinner, C. Vögele, J. Walter, and A. Wert. *Performance-oriented DevOps: A research agenda*. Technical Report SPEC-RG-2015-01. SPEC Research Group — DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), Aug. 2015.

[5] V. Ferme and C. Pautasso. "Integrating faban with docker for performance benchmarking". In: *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016)*. 2016, pages 129–130. DOI: 10.1145/2851553.2858676.

[6] C. Heger, A. van Hoorn, M. Mann, and D. Okanović. "Application performance management: State of the art and challenges for the future". In: *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2017)*. 2017, pages 429–432. DOI: 10.1145/3030207.3053674.

[7] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger. "Performance engineering for microservices: Research challenges and directions". In: *Companion of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2017)*. 2017, pages 223–226. DOI: 10.1145/3053600.3053653.

[8] A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Kiel Computer Science Series 2014/6. Dissertation, Faculty of Engineering, Kiel University. Kiel, Germany: Department of Computer Science, Kiel University, 2014. ISBN: 978-3-7357-5118-8.

[9] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[10] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske. "Hora: Architecture-aware online failure prediction". In: *Journal of Systems and Software* (2018). DOI: 10.1016/j.jss.2017.02.041.

[11] H. Schulz, T. Angerstein, and A. van Hoorn. "Towards Automating Representative Load Testing in Continuous Software Engineering". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. ICPE '18. Berlin, Germany: Association for Computing Machinery, 2018, pages 123–126. ISBN: 978-1-4503-5629-9. DOI: 10.1145/3185768.3186288.

[12] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, and H. Krcmar. "WESSBAS: Extraction of probabilistic workload specifications for load testing and performance prediction—A model-driven approach for session-based application systems". In: *Journal on Software and System Modeling (SoSyM)* (2016). DOI: 10.1007/s10270-016-0566-5.

[13] J. Walter, A. van Hoorn, H. Koziolek, D. Okanovic, and S. Kounev. "Asking "What"?, Automating the "How"? The Vision of Declarative Performance Engineering". In: *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*. ICPE '16. Delft, The Netherlands: Association for Computing Machinery, 2016, pages 91–94. ISBN: 978-1-4503-4080-9. DOI: 10.1145/2851553.2858662.

Automated Text Mining on Job Offers Using SAP HANA*

Analyzing Skill and Competency Requirements for Industry 4.0

Marlene Knigge, Sonja Hecht, Loina Prifti, and Helmut Krcmar

Technical University of Munich
Chair for Information Systems
{marlene.knigge,sonja.hecht,loina.prifti,krcmar}@in.tum.de

In this project, we extended the analysis of job offers in the field of Industry 4.0 we implemented in our former project.[1] Therefore, we used text analysis and text mining means provided by SAP HANA and the SAP Predictive Analytics Library (PAL). Industry 4.0 results in significant changes in our working environment. Employees need to be qualified to fulfill these needs. To analyze the skill and competency requirements towards employees in Industry 4.0, we collected job offers from two German online portals on a monthly base and applied text mining and text analysis to them. In this project, we improved our previously built application. We started the automation of the extraction of job offers from the internet and added a method for clustering the requirements in order to discover competency profiles emerging in the context of Industry 4.0.

1 Introduction

"Industrie 4.0"—or Industry 4.0—or the Industrial Internet enables disruptive solutions by combining known technologies in a new way: the Internet of Things (IoT), Smart Factories, Cyber Physical Systems (CPS), and the increased use of Embedded Systems [8, 10]. This development will influence the way we live—and the way we work, by enabling new ways of business value creation, which will result in changing business models and strategies, business processes and a change of the daily working life [10]. Employees need to be prepared to be able to fulfill new requirements [20]: "The requirements for the digitized skilled work will rise because the processes are interconnected and more complex, particularly with reference to the overlap of technical, organizational and social spheres of activity and the work process in the company" [6]. acatech et al. [13] describe in a study regarding competency development, that qualification is one of the essential factors in the Digital Transformation

*At this position, a different report was to appear—entitled "Applying Text Mining on Job Offers and Curricula Vitae Using SAP HANA". Unfortunately, it has erroneously published previously [12] in place of a previous report. Therefore, the previous report is presented here, instead.

The editors regret this error.

[1]For further information, please read the project report of our former project at HPI Future SOC Lab (Fall 2016) [11].

taking place in Germany. For this reason, identifying skills and competencies needed by employees working in an Industry 4.0 environment, becomes an important aspect as it can serve as base for tailoring training and (further) education—or the creation of new job profiles and HR strategies.

Our goal in this project is to address the need to get an overview of skills and competencies needed in an Industry 4.0 environment. We want to achieve this by extract relevant information from job offers in the area of Industry 4.0. Therefore, we collect job offers from German online job portals and then analyze them with SAP HANA text analysis and text mining.

2 Project Goal

The goal of our project is to extend the knowledge of the analysis of unstructured data from different sources— in this case: German online job offers. Taking into account our former project results (cf. [11]), we want to automate the extraction by implementing a web crawler. We want to extend and improve the analysis of extracted skills and competencies demanded from employees in an Industry 4.0 environment. We want to examine, which clustering algorithm is suited to analyze the extracted skills and competencies in order to discover new job profiles—and if this leads to meaningful results. Moreover, we want to compare of our results when the analysis is repeated on a monthly base.

We want to achieve this goal by applying methods and algorithms offered by SAP HANA and the SAP Predictive Analysis Library (PAL) to a data sample of German online job offers collected manually on a monthly base—whereby we want to automate this step of data collection.

3 Project Design

We build our work on a dataset, which was manually collected on a monthly base for eight months (cf. chapter 3.2). We process our data on SAP HANA and use algorithms provided by SAP PAL (cf. chapter 3.1). Additionally, we are working on automating the collection and loading of new job offers from open available online sources.

In a first subproject, we try to automate the step of collecting data from online job portals by implementing a web crawler and connecting it to our existing application.

Our second subproject concentrates on improving the extraction of skill and competency requirements from job offers and on analyzing, if we can derive job profiles from the data.

In our third subproject, we extract and analyze metadata comprised in the job offers and evaluate, if is suited to generate additional value to our former analysis.

3.1 System Configuration

The Hasso Platter Institute (HPI) provided us with a SAP HANA SPS12 with 1 TB RAM and 32 CPU cores. Additionally, we used the SAP HANA Predictive Analytics Library (PAL) and the Eclipse-based SAP HANA Studio, version 2.3.10. Our SAP HANA has been updated during this project phase to SAP HANA 2 SPS01. The implemented text analysis and text mining applications are executable in both configurations. For uploading the job offers, we implemented a script in Python version 3.5.2 using the module "pyhdb" version 0.3.2 (cf. figure 1).

The web crawler was implemented using node.js version 6.9.5 LTS, Long Term Support. This is available in SAP HANA from version 2.0 SPS 00 on—which was not available to us in the beginning of this project phase.

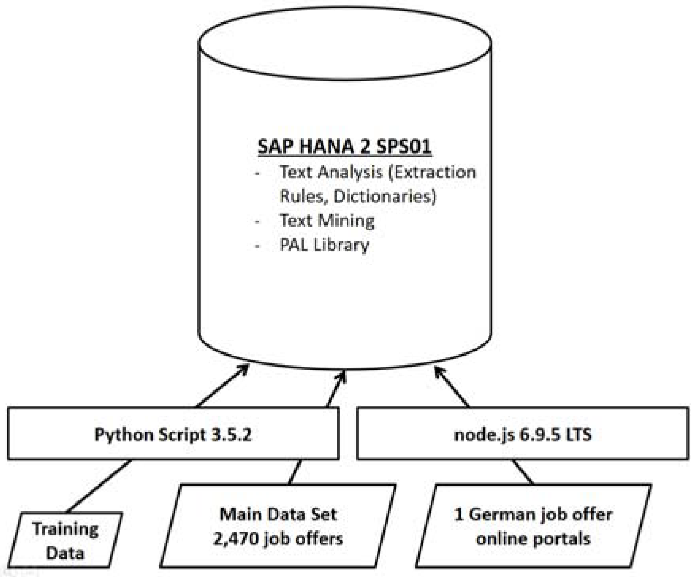

SAP HANA 2 SPS01
- Text Analysis (Extraction Rules, Dictionaries)
- Text Mining
- PAL Library
Python Script 3.5.2
node.js 6.9.5 LTS
Training Data
Main Data Set 2,470 job offers
1 German job offer online portals

Figure 1: IT architecture project for analyzing CVs (Own illustration)

3.2 Dataset

For extracting skill and competency requirements, we collected job offers manually on a monthly base over eight months (cf. table 1), whereat the job offers from job portal A have been saved as html-files, those from job portal B have been saved as pdfs. The overall dataset comprises 3,351 job offers collected manually from two different German online job portals. The job offers were found when applying the search term "Industrie 4.0". The analyses described in the following refer to a data set comprising the data from November 2016–April 2017, 2,470 job offers.

Table 1: Dataset: Manually collected job offers

	Job Portal A	Job Portal B	
Nov. 2016	178	100	
Dec. 2016	271	250	
Jan. 2017	210	175	
Feb. 2017	234	174	
Mar. 2017	231	176	
Apr. 2017	273	198	
Mai 2017	293	176	
Jun. 2017	237	175	**Total**
Total	1927	1424	**3351**

4 Web Crawler

After April, we started implementing a web crawler for being able to collect the job offers automatically. Our first approach was to implement the web crawler and the parser that is needed to process the data using XS Advanced, which supports Java and JavaScript/node.js development. SAP announced node.js as preferred programming language in the new XS Advanced environment [A]. Node.js is a JavaScript-based runtime environment, which makes the JavaScript engine V8 that is used in Google Chrome for example, available independently of any browsers [C].

As we wanted to use regular expressions, the V8 engine was better suited to serve our needs in comparison to the Java-VM [B]. Unfortunately, access to the XS Advanced was not stable enough to work with. However, XS Classic did not offer the functionalities we needed. Thus, we decided to implement the web crawler as standalone tool with an own command line interface. Currently, the web crawler is implemented using node.js version 6.9.5 LTS, Long Term Support. This is available in SAP HANA from version 2.0 SPS 00 on — which was not available to us in the beginning of this project phase.

It compares the job offers that are already available in SAP HANA to those it discovers on an online job portal by comparing the ID of the job offer given by the job portal. Thus, only duplicates from the same source are identified. If the same job offer appears on different job portals, it will appear in the result set several times as well. If the job offer is not contained in SAP HANA so far, it is downloaded, converted, and added to the dataset as html file and metadata such as URL, source (job portal) and an internal ID.

At present, we experiment with the current version of our web crawler mainly with one German online job portal. In a next step, we want to extend the use to more sources. As job offers from different sources tend to stick to different structures, we have to find out, how much adjustment is needed to get meaningful results when loading and analyzing job offers from different sources.

Listing 1: Full text Index for Text Analysis

```
CREATE FULLTEXT INDEX "JOBADV_IDX" ON
"SCHM"."STELLENANZEIGEN"("CONTENT")

TEXT ANALYSIS ON TEXT MINING ON SEARCH ONLY OFF FAST PREPROCESS OFF

LANGUAGE DETECTION('DE')

CONFIGURATION 'TA_SCHM::EXTRACTION_CORE_EXTENDED';

CREATE SYNONYM IDX FOR "SCHM"."$TA_JOBADV_IDX"
```

5 Skill and Competency Requirements Extraction

Due to parallel development of the web crawler and the skill and competency requirements extraction, we based the later on the manual extracted data set consisting of 2,470 job offers and described in chapter 3.2. In this project phase, we extended and refined our results from the former project phase (cf. [11]).

5.1 Data Pre-processing

After uploading the data set, we deleted duplicates from the whole data set as proposed by [2]. As it is not possible to compare elements with the datatype BLOB, we had to convert our job offers into BINARY first. In our data set of 2,470 job offers, we identified twelve duplicates. Next, we wanted to apply the SAP HANA FUZZY search to discover very similar job offers—which differ e. g. in the timestamp [16]. However, this is not working on BLOB objects. Thus, we needed to use text analysis for this step. We created a full text index using the configuration "LANGUAGE DETECTION 'DE' "—as our data set only contained German job offers. This requires the parameter "FAST PREPROCESS" to be disabled (cf. listing 1).

After applying the text analysis on our dataset, we discovered that 180 job offers were missing an entry in the full text index. When examining these job offers, we discovered that there was an error in the manual extraction: they were saved as pictures—and therefore not usable in our context. Above this, we could identify 210 job offers that were duplicates—including those identified in the first step described above.

5.2 Custom Dictionaries and Extraction Rules

In order to refine our text analysis and text mining results, we clustered the content we wanted to extract from the job offers in three areas: competencies, educational background, and experiences and knowledge. For each of these areas, we created custom dictionaries.

For competencies, we based it on the competency model of Erpenbeck [4] and one combined of that of Erpebeck [4] and Prifti et al. [14]. Next, we added synonyms extracted from the German online dictionary Duden [3]. After that, we used the LINGUISTIC search to find further variants of the competency terms [16]. Besides this, we added terms we found when explicitly searching for terms with typical characteristics of competencies, such as terms with the endings "-fähigkeit", "-kompetenz", or "-vermögen".

For the educational background, we implemented two dictionaries. One comprises terms dealing with the educational levels; the other one contains fields of studies. The later was based on the subject classification of the German Statistisches Bundesamt [19].

For the third area, experiences and knowledge, we created a fourth dictionary. This contains experiences, such as trainings. If knowledge items where close to competencies, they were integrated into the competencies dictionary.

The process of creating preferably complete dictionaries is a very labour intensive one. It is quite impossible to find each variant of the skills and competencies in focus. A second issue is that the extraction based on dictionaries faces shortcomings when analyzing enumerations. Here only the last term is identified correctly (e.g. "Team- und Kommunikationsfähigkeit"). For this reason, we used a second means to individualize the extraction process: custom extraction rules.

We created custom extraction rules using the "Custom Grouper User Language" (CGUL) and saving them as .hdbtextrule file in the SAP HANA repository. Custom extraction rules allow the use of stemming and part-of-speech (POS). e.g. with stemming, "is", "are" or "was" is matched to the stem "be". The POS of "is", "are" or "was" is verb.

We used our custom extraction rules in combination with our formerly created custom dictionaries in the areas of competencies and experiences and knowledge to improve our skills and competencies extraction. For the educational background, we did not implement custom extraction rules, as this would not have improved our results.

5.3 Evaluation

We did a manual classification of a subset of our data set to generate a training data set. Due to Jing [9], and Feldman and Sanger [5], manual classified data can be used for evaluating the results of information extraction systems. We chose the job offers for the training data set randomly.

The first evaluation was based on a training data set consisting on fifteen job offers. These contained 196 skill or competency requirements. Our application detected 126 of those (64 %). We had 15 (8 %) false positives. 39 (20 %) of the requirements have not been found, 16 (8 %) only partly. Thus, the precision is 89.4 %, the sensitivity 69.6 %. The F1-score is 78.3 %.

In the next step, we tried improve these values. One major problem were enumerations, especially with nouns when hyphens were used. To address this issue, we extended our extraction rules. Other reasons for negative results where spelling mistakes in the sources or mistakes resulting from the data pre-processing executed in SAP HANA before the analysis. In these cases it has to be decided how much effort should be made to extend custom dictionaries and custom extraction rules to find the last values.

A special problem occurred with the educational background as our custom dictionary contains the official terms. In job offers, sometimes individual names are used. These cases are not covered by our extraction mechanisms.

After adjusting our custom dictionaries and extraction rules, we did a second evaluation, based on the same training data set. With this, we were able to extract 73 % of the skill and competency requirements. We achieved a precision of 92.8 % and a sensitivity of 77.5 %. The F1-score is 84.5 %.

A second test on a new manually classified training data set which consist of fifty job offers lead to a total of 68 % extracted skill and competency requirements, a precision of 68 %, a sensitivity of 73 %, and an F1-score of 81 %.

5.4 Conclusion

Extracting skills and competencies from job offers using custom dictionaries and custom extraction rules is easy to understand and to maintain [18]. However, the implementation is costly and it only extracts requirements, which are implemented in the custom dictionaries or extraction rules [1].

Html-files needed further processing as they contain the source code of the website including invisible elements such as JavaScript code or meta-information. These items are considered when applying text analysis as well. It was needed to filter out those elements to improve our results.

6 Clustering of Skill and Competency Requirements

After we extracted skill and competency requirements from German online job offers in SAP HANA, we decided to analyze if we could find patterns in the requirements that might lead us to job profiles.

We decided to apply a frequency analysis, an association analysis, and clustering.

With the Frequency Analysis, the relative or absolute frequency of a word in a document collection or a subset of all documents is analyzed. In our case, we counted, in how many job offers a competency occurred. It was not taken into account if it occurred several times in one document. The most frequently mentioned requirements were languages (64 %), flexibility (40 %), capacity for teamwork (36 %), communication skills (29 %) and self-responsibility (25 %). This may be surprising as no technical skills can be found in the top five. However, we will need to bring these results together with our theoretical research on competencies for Industry 4.0 in the future.

The goal of the association analysis is to find values that occur together in the database. One method of the association analysis is the Apriori Algorithm, which extracts association rules [7, 15]. In the first step, we only examined rules that consisted of two competencies. We discovered 95 of those. As 87 of those contained one of the four most common requirements, this result is not too meaningful. Thus, we extended our analysis through combining rules with two competencies with further ones. We discovered ten competency groups. In a next step, we will have to interpret these results with our former theoretical research findings.

However, further analytics with different metrics will lead to different results in this area.

As done before with an association analysis, we tried out clustering for discovering competency groups. In this specific case, we applied the Agglomerate Hierarchical Clustering (AHC) algorithm. Here our result set contained 18 clusters. These comprise more competencies, which can be combined to competency profiles than it was the case with the Apriori algorithm.

For further analysis, we want to proceed with the results of the AHC algorithm.

7 Limitations

Our application is tailored to analyze German job offers. In the future, it could be extended to other languages, e.g. English. In this case, custom dictionaries and extraction rules will have to be adjusted to the additional language(s).

The current version of our web crawler is implemented as node.js standalone application with its own command line interface. Having SAP HANA version 2.0 SPS 01 available now, a next step could be to integrate this module into our SAP HANA application to avoid the need for switching the system.

Currently, we only use our web crawler on one German job portal. In a next step, we need to find out, how much adjustment is needed to allow crawling and processing from several sources to be able to access a broader data set.

8 Conclusions and Outlook

We implemented a web crawler for job offers in a first version, which is working in general. However, this needs further improvement and we have to establish a connection to our analysis implemented in SAP HANA.

We were able to extend our skill and competency extraction from job offers compared to our result in the former project phase. With around 90%, our precision is in line with those of other information extraction systems. The sensitivity of around 70% is in line with this. Only our F1-score of around 81% is lower than that of information extraction systems for established problems. For further improvement, we would recommend to extend the customer extraction rules and to convert some parts of the custom dictionaries into customer extraction rules whenever it is possible to recognize patterns. Maybe the Grammatical Role Analysis could help to reduce false positives. However, so far this tool is only available in English [17].

We tried out three methods in order to discover patterns that may lead us to competency profiles for Industry 4.0: frequency analysis, classification analysis, and clustering. In this case, clustering with the AHC algorithm lead to the most promising results.

In a follow-up project, we want to finish the automation of our whole application, starting from the extraction of the job offers, over the pre-processing, to the

skill and requirement extraction. This will allow us to execute more complex analyses on our dataset, and include analysis regarding changes in skill and competency requirements over time.

In the previous project phase, we started to extract metadata from job offers using custom dictionaries and custom extraction rules. Another step in the future may be to resume this idea, improve the extraction and analyze the metadata together with our job profiles.

Acknowledgments

We would like to thank our master student Sebastian Hagn for being part of our project and for his dedication and ideas. Moreover, we wish to thank our student assistants Leon Kevin Birkel and Tristan Hopf for their support.

References

[1] F. Bensberg and G. Buscher. "Digitale Transformation und IT-Zukunftsthemen im Spiegel des Arbeitsmarkts für IT-Berater – Ergebnisse einer explorativen Stellenanzeigenanalyse". In: *Multikonferenz Wirtschaftsinformatik (MKWI) 2016, Ilmenau*. 2017, pages 1–12.

[2] F. Bensberg and G. Buscher. "Job Mining als Analyseinstrument für das Human-Resource-Management". In: *HMD Praxis der Wirtschaftsinformatik* 53.6 (2016), pages 815–827. DOI: 10.1365/s40702-016-0256-3.

[3] *Duden*. 2017. URL: http://www.duden.de/ (last accessed 2017-03-13).

[4] J. Erpenbeck. *Handbuch Kompetenzmessung: erkennen, verstehen und bewerten von Kompetenzen in der betrieblichen, pädagogischen und psychologischen Praxis*. Schäfer-Poeschel, Stuttgart, Germany, 2007. DOI: 10.34156/9783791035123.

[5] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge, MA, USA: Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546914.

[6] J. Gebhardt, A. Grimm, and L. M. Neugebauer. *Developments 4.0 - Prospects on future requirements and impacts on work and vocational education*. Volume 2015. 3. Journal of Technical Education, 2015, pages 117–144.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2nd edition. New York, NY, USA: Springer, 2009. DOI: 10.1007/978-0-387-84858-7.

[8] *Industrie 4.0: Die neue Hightech Strategie – Innovationen für Deutschland*. 2016. URL: http://www/hightech-strategie.de/_dpsearch/highlight/searchresult.php?URL=http://www.hightech-strategie.de/Industrie-4-0-999.php&QUERY=industrie+4.0 (last accessed 2016-03-09).

[9] J. Jing. "Information Extraction from Text". In: *Mining Text Data. Springer, New York, NY, USA*. Edited by C. C. Aggarwal and C. X. Zhai. 2012, pages 11–42. DOI: 10.1007/978-1-4614-3223-4_2.

[10] H. Kagermann, W. Wahlster, and J. Helbig. *Securing the future of German manufacturing industrie – Recommendations for implementing the strategic initiative INDUSTRIE 4.0*. Final report of the Industrie 4.0 Working group. 2013.

[11] M. Knigge, L. Prifti, S. Hecht, and H. Krcmar. "Text Mining on Job Offers Using SAP HANA: Analyzing Skill and Competency Requirements for Industry 4.0". In: (2017). Edited by C. Meinel, A. Polze, K. Beins, R. Strotmann, U. Seibold, K. Rödszus, and J. Müller. In press.

[12] M. Knigge, T. Neumer, F. Willnecker, and H. Krcmar. "Applying Text Mining on Job Offers and Curricula Vitae Using SAP HANA. Skills and Competencies for Industry 4.0". In: *HPI Future SOC Lab – Proceedings 2017*. Edited by C. Meinel, A. Polze, K. Beins, R. Strotmann, U. Seibold, K. Rödszus, and J. Müller. Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam. Universitätsverlag Potsdam. ISBN: 978-3-86956-475-3. DOI: 10.25932/publishup-43310.

[13] *Kompetenzentwicklungsstudie Industrie 4.0*. acatech, Fraunhofer IML, and equeo GmbH, 2016.

[14] L. Prifti, M. Knigge, H. Kienegger, and H. Krcmar. "A Competency Model for "Industrie 4.0" Employees". In: *Proceedings of the 13th International Conference on Wirtschaftsinformatik (WI) 2017, St. Gallen, Switzerland*. 2017, pages 40–60.

[15] *SAP HANA Predictive Analysis Library*. SAP SE. 2017. URL: https://help.sap.com/hana/sap_hana_predictive_analysis_library_pal_en.pdf (last accessed 2017-10-24).

[16] *SAP HANA Search Developer Guide*. SAP SE. 2017. URL: https://help.sap.com/hana/SAP_HANA_Search_Developer_Guide_en.pdf (last accessed 2017-10-24).

[17] *SAP HANA Text Analysis Language Reference Guide*. SAP SE. 2017. URL: https://help.sap.com/doc/2e76b520f80e4fb0b4c91a756f5f51f7/2.0.01/en-US/SAP_HANA_Text_Analysis_Language_Reference_Guide_en.pdf (last accessed 2017-10-24).

[18] S. Sarawagi. "Information Extraction". In: *Foundation and Trends in Databases* 1.3 (2007), pages 261–377. DOI: 10.1561/1900000003.

[19] Statistisches Bundesamt. *Bildung und Kultur. Studierende an Hochschulen – Fächsersystematik*. Wiesbaden, 2015. URL: https://www.destatis.de/DE/Methoden/Klassifikationen/BildungKultur/StudentenPruefungsstatistik.pdf (last accessed 2017-10-24).

[20] Zukunftsprojekt Industrie 4.0. *Digitale Wirtschaft und Gesellschaft*. 2016. URL: http://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html (last accessed 2016-03-09).

Global-Scale Internet Graphs

Extended Vulnerability Analyses

Benjamin Fabian[1,2], Tatiana Ermakova[3], Stefan Kelkel[2], and Annika Baumann[2]

[1] Hochschule für Telekommunikation Leipzig
[2] Humboldt-Universität zu Berlin, Germany
bfabian@wiwi.hu-berlin.de, annika.baumann@wiwi.hu-berlin.de, stefan.kelkel@googlemail.com
[3] Universität Potsdam, Germany
TErmakova@zi.de

Based on BGP and traceroute data integrated from global-scale mapping projects, we analyze the Internet graph with respect to connectivity and robustness. In an evolution of the initial project, we assess several malware strategies that could affect border routers and their corresponding Autonomous Systems.

In this project phase, we consolidated earlier results by repeated experiments, and started to extend our study to the connectivity of financial services.

1 Introduction

Our research project [1, 2, 3, 4, 8, 9, 10, 12] aims at developing methods for creating and analyzing a large integrated set of Internet graphs at several abstraction levels.

This serves as a basis for our ongoing analyses searching for bottlenecks and weak points in the entire Internet topology as well as in the connectivity of individual firms and services.

As our project evolved, a novel line of research studies attack strategies motivated by autonomic malware that could affect important border routers, and investigates their impact on Internet robustness via graphbased simulations.

In the current project phase, we made our earlier results more robust by repeating simulations that involved random target selection.

Furthermore, we began an investigation of connectivity of financial services.

2 Research Approach

Based on graphs constructed from empirical routing data, we simulate destructive strategies for Internet worms that would affect border routers and study their effects on global Internet connectivity on the Internet Protocol (IP) and Autonomous Systems (AS) levels.

3 Related Publications

Our central framework CORIA for making the results of such graph analyses useful in practical applications was presented at IEEE ICC 2017 in Paris [6]. It is currently under further development with respect to multiple features [13].

Further publications based on the current project have been accepted [5, 15] or are under development [7, 14]. We list some of them as technical reports but note that some of them are not finalized, respectively, currently under review.

4 Project Plan

Our project requires powerful computation capabilities based on the large-scale memory and multicore architecture of the HP Converged Cloud and, potentially, also the newly implemented SAP HANA Graph Engine.

This project is structured in several phases. The current phases of our project study the destructive capabilities of malware spreads via outage simulations. This requires the repetition of experiments that

involve random target selection in order to achieve stronger statistical significance.

With the help of the computational power of HPI Future SOC Lab, we are much better equipped to examine graph measures such as centrality metrics, clustering coefficients, shortest paths, and connected components.

The novel project phases will also be carried out on the resources provided by the HPI Future SOC Lab [11].

5 Project Status and Results

5.1 Simulations of Attacks Inspired by Malware

We simulate attacks inspired by malware threats. A systematic vulnerability assessment of the Internet's core components would help to identify critical areas and to build more resilient structures.

In particular, the border routers, which are interconnecting the autonomous systems, could constitute critical bottlenecks. Specifically, we aim to quantify the impact of attacking border routers at the autonomous system level.

For this purpose, various worm strategies are simulated.

These worms are assumed to infiltrate and shut down he router operating systems of leading suppliers, such as Cisco and Juniper.

In order to identify these router operating systems, a TTL-based fingerprinting method is conducted.

The results for different attack scenarios are compared and analyzed. In order to increase statistical significance, several simulation runs needed to be repeated in the current project phase.

5.1.1 Preliminary Results

In this malware strategy analysis, we will base on the so-called Ark ITDK AS graph, which will be discussed in [7]. This graph comprises 38,417 nodes and more than 230,000 edges. We observed that this graph exhibits *Small World* characteristics.

A novel example of the simulations we conducted is shown in figure 1, which in contrast to the earlier report [10] is now based on averaging several repetitions of the random strategies and shows a different metric for analysis.

Here, four AS-eclipsing spreading strategies are studied with respect to their impact on the average node degree. This figure shows massive breakdowns of connectivity after a certain threshold of removed routers has been reached.

Further details will be presented in [7], which is currently still in submission.

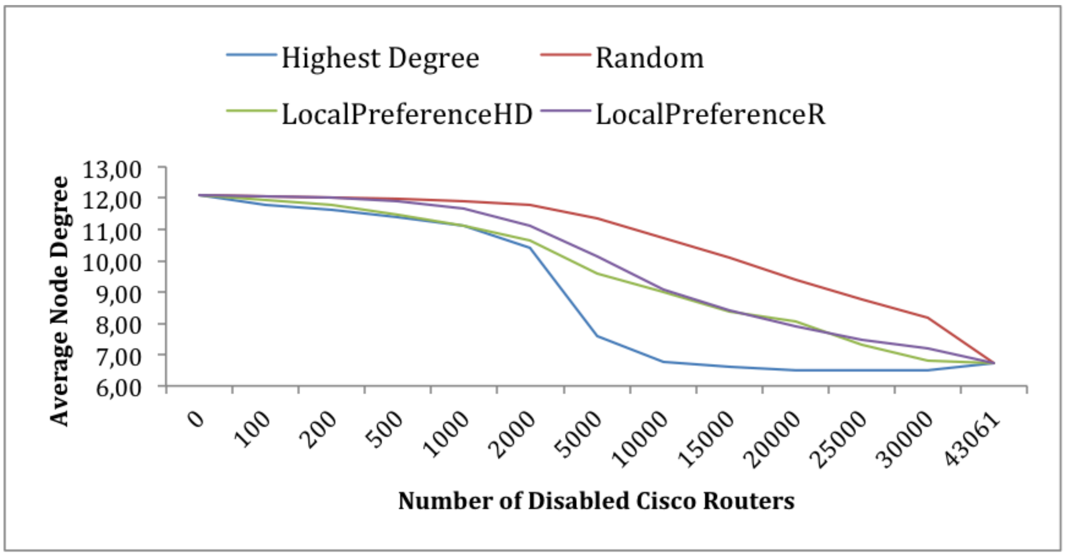


Figure 1: Development of the Average AS Node Degree (Attacks on Cisco Routers)

5.1.2 Use of Hardware Resources

The hardware provided by the HPI Future SOC Lab so far included three HP Converged Cloud Blades with 24 x 64-bit CPUs running at a frequency of 1.2 GHz on Ubuntu. Each of the three machines had 64 GiB of memory and was equipped with 1 TiB HDD. This configuration offers an extensive parallelization of tasks.

The calculated results would not have been possible without the support of the HPI Future SOC Lab [11]. Due to the intensive calculations necessary for the vulnerability analyses, the use of HPI Future SOC Lab resources has been very important for years.

6 Conclusion

In future work, we aim to conduct further vulnerability analyses based on the extensive data sets that we have collected so far. One promising direction is to further study the connectivity of specific business services. All of our future activities will require the massive calculation capacities enabled through HPI Future SOC Lab resources. We are very grateful for the continuing support of our project.

References

[1] A. Baumann and B. Fabian. "How Robust is the Internet? – Insights from Graph Analysis". In: *Proceedings of the 9th International Conference on Risks and Security of Internet and Systems (CRiSIS 2014), Trento, Italy, Springer, LNCS 8924*. 2014. DOI: 10.1007/978-3-319-17127-2_18.

[2] A. Baumann and B. Fabian. "Towards Measuring the Geographic and Political Resilience of the Internet". In: *International Journal of Networking and Virtual Organisations* 13.4 (2013), pages 365–384. DOI: 10.1504/IJNVO.2013.064465.

[3] A. Baumann and B. Fabian. "Vulnerability Against Internet Disruptions – A Graph-based Perspective". In: *Proceedings of the 10th International Conference on Critical Information Infrastructures Security (CRITIS 2015), Berlin, Germany*. Oct. 2015. DOI: 10.1007/978-3-319-33331-1_10.

[4] A. Baumann and B. Fabian. "Who Runs the Internet? Classifying Autonomous Systems into Industries". In: *Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST), Barcelona, Spain*. Apr. 2014.

[5] S. Dombrowski and B. F. Tatiana Ermakova. "Graph-Based Analysis of Cloud Connectivity at the Internet Protocol Level". In: *International Journal of Communication Networks and Distributed Systems* (2018). accepted.

[6] B. Fabian, A. Baumann, M. Ehlert, T. Ermakova, and V. Ververis. "CORIA – Analyzing Internet Connectivity Risks Using Network Graphs". In: *Proceedings IEEE International Conference on Communications (ICC 2017), Paris, France*. 2017. DOI: 10.1109/ICC.2017.7996828.

[7] B. Fabian, A. Baumann, T. Ermakova, and S. Kelkel. *Internet Robustness Analysis – Simulation of Worm-Based Router Attacks*. Working Paper. 2017.

[8] B. Fabian, A. Baumann, and J. Lackner. "Topological Analysis of Cloud Service Connectivity". In: *Computers & Industrial Engineering* 88 (Oct. 2015), pages 151–165. DOI: 10.1016/j.cie.2015.06.009.

[9] B. Fabian and G. Tilch. *Analyzing the Global-Scale Internet Graph at Different Topology Levels: Data Collection and Integration*. HPI Future SOC Lab Day Workshop & Report. 2015.

[10] B. Fabian, A. Baumann, and T. Ermakova. *Stefan Kelkel: Global-Scale Internet Graphs: Vulnerability Analysis Based on Worm-Spread Simulations*. Technical Report. HPI Future SOC Lab Project Report, Apr. 2017. DOI: 10.13140/RG.2.2.10905.01127.

[11] *HPI Future SOC Lab*. URL: https://hpi.de/forschung/future-soc-lab.html (last accessed 2018-03-27).

[12] M. Huth and B. Fabian. "Inferring Business Relationships in the Internet Backbone". In: *International Journal of Networking and Virtual Organisations* (2016). DOI: 10.1504/IJNVO.2016.081651.

[13] D. Kiene-Maksimovic, E. Zinovyeva, B. Fabian, and J. Park. "CORIA 2.0 Augmenting a Universal Framework for Connectivity Risk Analysis". In: *SSRN Electronic Journal* (Oct. 2017). DOI: 10.2139/ssrn.3060327.

[14] G. Tilch, T. Ermakova, and B. Fabian. *A Multilayer Graph Model Of The Internet Topology*. Working Paper. Oct. 2017. DOI: 10.5281/ZENODO.1038599.

[15] U. Yacobi-Keller, E. Savin, B. Fabian, and T. Ermakova. "Towards Geographical Analysis of the Autonomous System Network". In: *International Journal of Networking and Virtual Organizations* (2018). accepted.

Towards a NUMA-aware Task-based Fast Multipole Method

Laura Morgenstern

Operating Systems Group
Faculty of Computer Science
Chemnitz University of Technology
lmor@hrz.tu-chemnitz.de

Molecular dynamics simulations have become a vital research method in biochemistry and materials science. In molecular dynamics, the fast multipole method is used to reduce the complexity of the computation of pairwise long-range interactions. Since the computation of long-range interactions is the most expensive part of molecular dynamics [4], its efficiency and scalability is decisive for the whole simulation. The goal of this project is to analyze the scalability of the fast multipole method on modern multicore systems. Today's multi- and manycore systems are based on NUMA (Non-Uniform Memory Access) architectures for reasons of scalability. To exploit the scalability of the hardware, latency- and bandwidth-critical applications have to take data locality into account. This holds especially when aiming for strong scaling like modern molecular dynamics simulations do. We outline the NUMA-aware data placement and load balancing policies presented in [5] in Sections 2 and 3. The detailed performance analysis provided in Section 4 reveals that NUMA-awareness increases the performance of a state-of-the-art implementation of the fast multipole method by a factor of 2.43 on eight NUMA nodes.

1 Introduction

1.1 Non-uniform Memory Access

Non-uniform memory access (NUMA) is a shared memory model used in modern multiprocessor systems. A multiprocessor system with NUMA architecture consists of several NUMA nodes. Each NUMA node is a set of cores around local memory which the cores can directly access. The cores of each NUMA node can access the memory of remote NUMA nodes by traversing the NUMA interconnect. In comparison to accessing the local memory, this exhibits higher latency and lower bandwidth. To exploit highly scalable and massively parallel NUMA architectures, software needs to adapt to these differences in latency and bandwidth.

1.2 Task-based Fast Multipole Method in a Nutshell

The fast multipole method (FMM) is a fast summation method that is used to compute all pairwise interactions in molecular dynamics simulations. It reduces the computational complexity of classical coulomb solvers from $O(N^2)$ to $O(N)$. This achievement is based on the idea that distant particles affect an observed particle less than particles in the proximity. However, the influence of the distant particles is not zero. Hence, it cannot be neglected when searching for an accurate solution. To exploit this idea, a hierarchical space decomposition is used to define groups of more and more particles with increasing distances between those groups. Particles belonging to different and not neighboring groups interact in the far field. Particles belonging to the same group and particles from neighboring groups interact in the near field. To perform a hierarchical space decomposition, the simulation box is bisected in each dimension. By recursively continuing this principle, a tree structure develops. The workflow of the FMM can be described along this tree structure. For algorithmic details on the workflow of the FMM, please refer to [3]. Considering a task-based FMM, the tree structure can be seen as a task graph of the workflow exhibiting horizontal and vertical dependencies between the data elements of the tree. Since these dependencies require data transfer between threads running on different NUMA nodes, we aim to reduce inter-node data transfer by increasing data locality. For the performance analysis we use *FMSolvr*, a state-of-the-art `C++`-implementation of a task-based FMM, as test application.

2 NUMA-aware Data and Thread Placement

To distribute the workload as equally as possible, the assignment of data to threads is realized through an equal partitioning of the number of tasks on each tree level. Based thereon, data locality is assured by placing a thread and the data it works on the same NUMA node. Sticking to these principles, we present the data placement and thread pinning policies *Scatter Principally* (SP), *Compact Ideally* (CI) and *Compact Scatter* (CS).

When using SP, a classical load-balancing approach is applied. In general case this means that all NUMA nodes are used, with an equal amount of threads being assigned to each NUMA node. Using CI means in turn that as few as possible NUMA nodes are used. This is implemented by assigning threads to a single NUMA node as long as the NUMA node has got idle cores. Not before all non-SMT (Simultaneous Multi-Threading) cores of a NUMA node are busy, the next NUMA node is filled up with threads. CS is an experimental combination of SP and CI that serves the analysis of NUMA dependent on SMT.

3 NUMA-aware Load Balancing

Work stealing is a load-balancing approach for task-based applications. As soon as it runs out of work, a thread may steal and execute tasks from other threads. Provided that the overhead for stealing tasks is sufficiently low, this leads to a shorter time to solution. We refer to the default load balancing approach as *ALL*, since a thread may

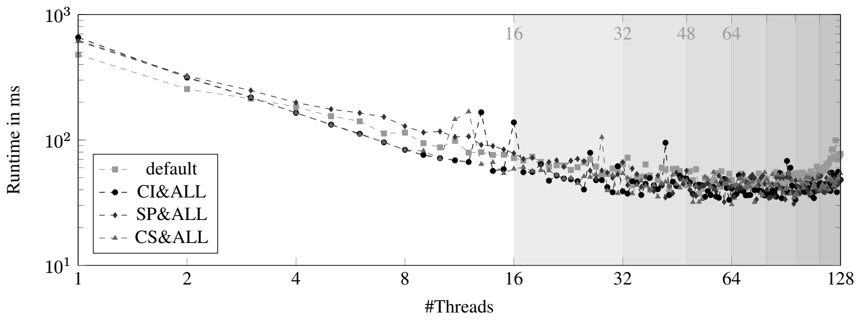


Figure 1: Runtime with NUMA-aware data placement policies CI, SP and CS with default load balancing policy ALL

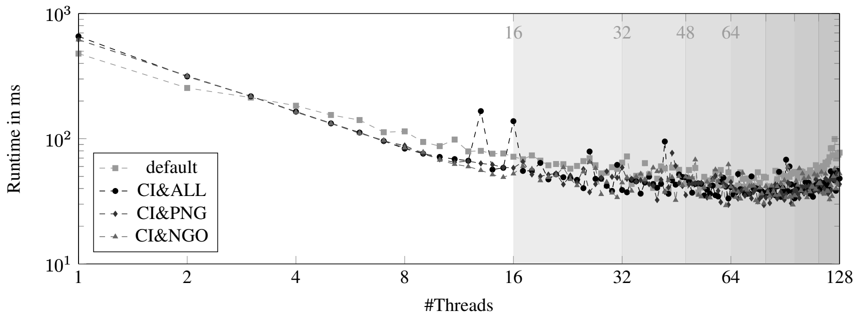


Figure 2: Runtime with load balancing policies ALL, PNG and NGO

steal work from all the other threads. In NUMA systems this overhead depends on the location of the thread relative to the location of the task it attempts to steal. To analyze the trade-off between load-balancing and NUMA-awareness, we present the work stealing policies *Prefer NUMA-Group* (PNG) and *NUMA-Group Only* (NGO).

When using PNG, a thread tries to steal work from another thread located on the same NUMA-node first. If there is no work to be stolen, the thread tries to steal work from threads on remote NUMA nodes with increasing distance to the local NUMA-node. When using NGO, a thread steals work from the local NUMA-node only. If there is no work to be stolen, the thread stays idle.

4 Performance Analysis

4.1 Hardware

The performance of FMSolvr is examined on one of the Future SOC Lab's multicore servers with eight NUMA nodes, with each NUMA node being an Intel Xeon X7560 with 8 cores and 2-way SMT (Simultaneous Multithreading). Hence, the measurements are performed on up to 128 SMT-cores.

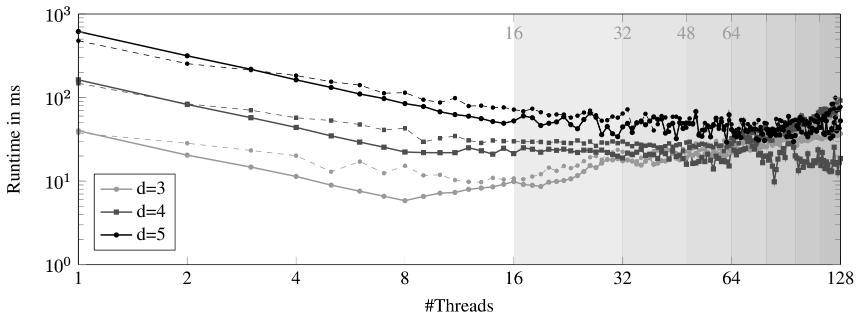


Figure 3: Runtime with data placement policy CI and load balancing policy NGO for FMM tree depths $d = 3, \ldots, 5$

4.2 Input Data Set

The input data set is a homogeneous particle ensemble with 1000 particles. The values of positions and charges are random numbers in range $[0, 1]$. However, the runtime of FMSolvr does not only depend on the input data set, but also on the FMM-parameters described in Section 1.2. The measurements are performed with multipoleorder $p = 3$. This leads to relatively low computational effort per compute node and hence supports strong scaling. The FMM tree depth d is explicitly stated for each measurement.

4.3 Measuring Techniques

To get stable timing results, we arithmetically average the runtimes of 100 executions of FMSolvr with the input data set described in Section 4.2. Thereby, each execution covers the whole workflow of the FMM for a single time step of the simulation. During the measurements, Intels Turbo Boost is disabled, since it leads to varying clock frequencies depending on the workload and the number of used cores. Hence, scaling plots may be distorted since the clock frequency is automatically increased for the single-thread implementation due to it using one core only. As stated in Charles, Jassi et al. [1], the Turbo Boost feature may on average lead to a 6 % reduction in execution time for computationally-intensive programs by accelerating sequential phases. Furthermore, we use *jemalloc*, a general purpose malloc implementation that emphasizes fragmentation avoidance and scalable concurrency support [2], to support scalable memory allocation.

4.4 NUMA-aware Data and Thread Placement

Figure 1 shows the runtime of FMSolvr for tree depth $d = 5$ with different NUMA-aware data and thread placement policies. In the majority of measurements, the NUMA-awareness policies CI and CS outperform the default version of FMSolvr. With NUMA-aware data and thread placement, FMSolvr is up to 2.14 times faster. The maximum performance improvement of 2.14 is reached for 125 threads. However, there are occasionally unfavorable runtime peaks for CI and CS. These peaks are due to SMT, since they are reproducible and don't occur for policy SP, which avoids using SMT for $\#Threads \leq 64$. Even though SP improves data locality, it usually performs worse than the default version for $\#Threads < 64$. This is due to the increased amount of inter-node data transfers and shows that classical load balancing may not be the right choice for large NUMA systems.

4.5 NUMA-aware Load Balancing

Figure 2 shows the runtime of FMSolvr for tree depth $d = 5$ with data placement policy CI with different load balancing policies. Equipped with NUMA-aware data placement and NUMA-aware load balancing FMSolvr is up to 2.43 times faster. Furthermore, NUMA-aware load balancing partly compensates the former SMT-induced runtime peaks. Since PNG and NGO outperform ALL, load should preferably be balanced between threads running on the same NUMA node only.

4.6 NUMA-awareness Dependent on Tree Depth

Figure 3 shows the runtime of FMSolvr with data placement policy CI and load balancing policy NGO in comparison to the default version (dashed line) for FMM tree depths $d = 3, \ldots, 5$. As can be seen best for $\#Threads = 8$, the smaller the depth d of the FMM tree, the larger the distance of the related runtime plots. Hence, the impact of the NUMA architecture on runtime increases with decreasing tree depth. For $\#Threads = 3$, the computational effort is so low that using cores located on remote NUMA nodes even increases runtime. However, enabling strong scaling requires further reduction of the computational effort per compute node. Hence, we need to reduce the data transfer between NUMA nodes further.

5 Conclusion and Future Work

The results of the performance analysis show that NUMA-aware data and thread placement in combination with NUMA-aware load balancing speeds up a single time step of the FMM by a factor of 2.43. This implies that NUMA-awareness is crucial for latency-critical molecular dynamics simulations — especially on large NUMA systems. However, FMSolvr does still not scale for input data sets exhibiting very low computational effort per compute node. To further improve scaling through NUMA-awareness, the following is next up on our agenda:

- Develop a generic NUMA-aware pool allocator,
- Develop a less strict thread pinning policy that avoids using SMT,
- Reduce data transfer between NUMA nodes further, e.g. through data replication or a more regular tree partitioning.

References

[1] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova. “Evaluation of the Intel® Core™ i7 Turbo Boost feature”. In: *2009 IEEE International Symposium on Workload Characterization (IISWC)*. 2009, pages 188–197. DOI: 10.1109/IISWC.2009.5306782.

[2] J. Evans. *jemalloc memory allocator*. URL: http://jemalloc.net (last accessed 2018-03-07).

[3] I. Kabadshow. *Periodic Boundary Conditions and the Error-Controlled Fast Multipole Method*. Schriften des Forschungszentrums Jülich. Forschungszentrum Jülich GmbH, 2012. ISBN: 978-3-89336-770-2.

[4] I. Kabadshow, H. Dachsel, C. Kutzner, and T. Ullmann. “GROMEX – Unified Long-range Electrostatics and Flexible Ionization”. In: *Innovatives Supercomputing in Deutschland* 11.2 (2013), pages 72–75. HDL: 2128/17846.

[5] L. Morgenstern. *A NUMA-Aware Task-Based Load- Balancing Scheme for the Fast Multipole Method*. Mas- ter’s thesis, Chemnitz University of Technology, 2017.

Introducing NUMA-awareness to the SIFT Algorithm

Max Plauth, Felix Eberhard, Felix Wolff, and Andreas Polze

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

In the *Fall 2017* period, we continued our work on investigating the behavior of large NUMA systems for data-intensive workloads. Building up on previous work on both a NUMA-aware implementation of the SIFT algorithm and PGASUS, a prototypical library for NUMA-aware programming in C++, we integrated both projects in order to evaluate the performance of PGASUS in a real-world use case.

1 Introduction

Building up on top of previous research done on the adaption of the *Scale-Invariant Feature Transform* (SIFT) algorithm for *Non-Uniform Memory Access* (NUMA) systems using *OpenMP*, we implemented task parallelism and data locality in SIFT using a prototypical C++ programming model, *PGASUS*. We did so to replace a makeshift construct to enforce locality of data and work which was introduced because *OpenMP* does not provide good locality constructs. *PGASUS* is the result of a masters thesis done at the Operating Systems and Middleware Group at the Hasso Plattner Institute [1].

We want to start by motivating the problem in section 2 where we explain the connection drawn between *NUMA*, *SIFT* and *PGASUS*. Since we built on existing and barely documented code, we will provide such a documentation along with notes on the *PGASUS* -based SIFT-implementation in section 3. Finally, we conclude this report with section 4.

2 Motivation

This Section will give an in-depth explanation of relevant details in the inner workings of *NUMA*, *SIFT* and how *PGASUS* fits in.

2.1 NUMA: colocation of data and work

NUMA systems are organized in nodes, where each node has memory that is connected to it. The nodes themselves are connected as well, as figure 1 illustrates.

From the illustration, it becomes apparent that node S1 may access memory on node S0, but will have to pay the premium of increased latency. Memory access times vary depending on location, hence the name of the architecture. Thus, the ultimate

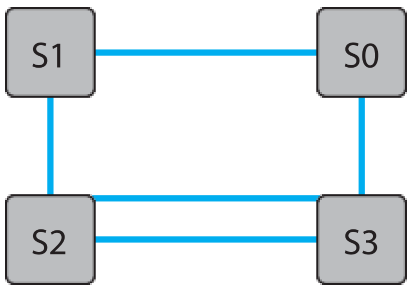


Figure 1: An illustration of typical 4-socket NUMA topology

goal for applications running on *NUMA* systems should be the highest possible co-location of data and work to minimize avoidable waiting times.

2.2 The SIFT Algorithm

The *Scale-Invariant Feature Transform* (SIFT) algorithm is popular in computer vision [2] as it allows object recognition without prior objectpreparation with markers. *SIFT* also has similarities to current big-data algorithms in that it uses stencil operations and similar memory access-patterns [3]. Taking a single image as an input, the algorithm builds differently sized copies of the image. Each level of resolution is called an *octave*, a term that is of key importance to the following work. The resolution of an octave is always half of that before it.

Inside each octave, the image is again multiplied, but left of the same size. Gauss-filters of varying strengths are applied to the different copies inside an octave, as illustrated in figure 2.

The processing that follows is the same for every octave. After the application of the Gauss filters, several new images are created containing the *difference of Gaussians* (*DoG*) for the different levels of each octave. Having k blurred images in an octave results in $k-1$ DoG images. These are then further condensed into images representing maxima and minima over three images. The aforementioned stencil operation takes place at this stage. The *SIFT* author David Lowe recommends generating two extrema images per octave, leading to a minimum of 5 blurred images per octave.

After checking the extrema for certain criteria, key-points are generated and brought together with the key-points from the other octaves. This unification of results concludes the algorithm. A well-documented walk-through of the algorithm can be found on aishack.in.

2.3 SIFT on NUMA

In previous work, we used *OpenMP* and *libnuma* to approximate data-locality [3]. With octaves being objects that implement the steps described above, every octave places a data object on every node. An octave object does so by opening up an *OpenMP* parallel section in which each thread checks for the node that it is running

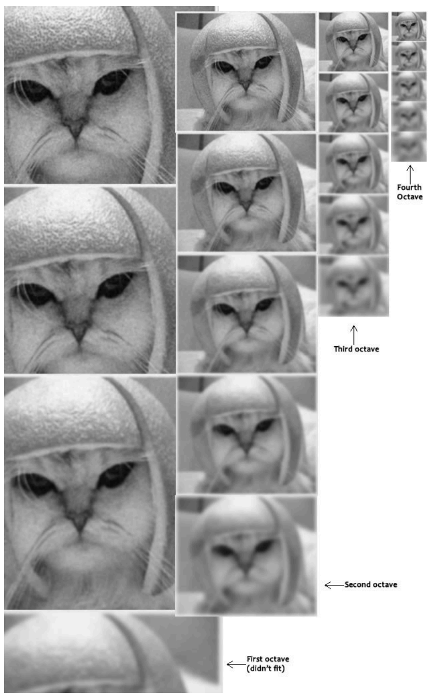


Figure 2: An illustration of octaves, their sizes and varying Gauss filter levels

on via *libnuma*. Listing 1 shows the relevant code snippets from the initialization section of the octave.

Thus, when data-locality is required, each thread only needs to find out which node it is working on and use its node id as an index to the `nodeData` array above.

2.4 The case for PGASUS

The aforementioned approach has been tested and verified to bring significant speed improvements [3]. However, it has a fundamental flaw. *OpenMP* does not guarantee thread locality nor is the data accessed guaranteed to be local. This is because Linux may remap pages when memory layout changes appear beneficial, thus diminishing the usefulness of the `nodeData` array approach. The approach relies heavily on the inactivity of the operating system which can't be provided.

PGASUS is well suited to eliminate this flaw. As listing 2 shows, *PGASUS* natively supports both the data- and task-locality paradigms without workarounds. The example illustrates the creation of *Memory Sources*, objects which guarantee that memory used for the creation of new objects is local to a specific node.

Afterwards, a thread that should execute a task called *taskOnNodeMemory* on a specific node is created. *PGASUS* guarantees that this thread will only be run on the specified node. Every instance of the function activates a *PlaceGuard*, an object that will source any memory allocation call from the specified memory source. Thus the created vector will be created from memory local to the node that the memory source

Listing 1: Implementing data-locality on nodes via OpenMP and libnuma

```
// abridged from void NumaOctave::setupNodeData(int maxNodes)
#pragma omp parallel num_threads(totalThread Count)
{
    int numaNodeId = getNumaNodeID();
    Thread Group Descriptor tgd = tgp.checkin();

    // create per node data structures
    if ( tgd. thread_num == 0) {
        assert(nodeData[numaNodeId] == NULL);
        nodeData[numaNodeId] = new NodeData ();
    }
}
```

lies on. As specified before, this is also the node that the thread is being executed on. Thus, thread- and data locality have been achieved.

3 Adapting an existing SIFT implementation

In this Section we want to document our learnings about the code and give insights into the adaptions that were taken to implement a *PGASUS*-enabled *SIFT*. In the following sections we will highlight two C++ classes: `NumaOctave` and `PgasusOctave`.

The first represents the aforementioned attempt at bringing *NUMA*-awareness to *SIFT* via *OpenMP*. The latter represents our approach with the *PGASUS* programming model.

3.1 Class hierarchy

Octaves are central to *SIFT* and so they are to this implementation. Building upon the `struct SiftParams`, the class `SiftOctaveBase` is the base of both `NumaOctave` and `PgasusOctave`. The hierarchy is visualized in figure 3.

`SiftParams` parses command-line arguments and sets default values for variables relevant to calculations. The base class `SiftOctaveBase` delivers a framework that implements a write-optimized Gaussian filter. Its constructor receives an image of the corresponding scale (see section 2) and an index that indictates the octave number.

All classes described in the following paragraph inherit from `SiftOctaveBase`. `SimpleOctave` is a non-parallel implementation of *SIFT*, whereas `NumaOctave` is the *OpenMP*-enabled and NUMA-awareness-approximating version discussed above. The class `PgasusOctave` represents the *PGASUS*-enabled version without *OpenMP*.

For benchmarking purposes, `NumaOctave` has been serialized to match the amount of parallelism in `PgasusOctave`. This version is called `NumaOctavePg`.

Listing 2: The PGASUS approach to implementing data- and task-locality

```
int main(int argc, char const* argv []) {
  int priority = 0;
  list <numa::TriggerableRef> taskList;
  vector <numa::Mem Source> msList;
  numa::NodeList nodeList = numa::NodeList::allNodes();

  for (int i = 0; i < nodeList. size(); ++i) {
    numa::Mem Source ms = numa::MemSource::create(
      nodeList[i], 1000, "memory_source");
    msList.push_back(ms);

    auto task On NodeMemory = [&](){
      numa::PlaceGuard g(ms);
      new vector <string> vectorIn Memory Source ;
    }

    taskList. push_back(
      numa::async <void>(taskOnNodeMemory,
                         priority,
                         nodeList[i]));
  }

  numa::wait(taskList);

  return 0;
}
```

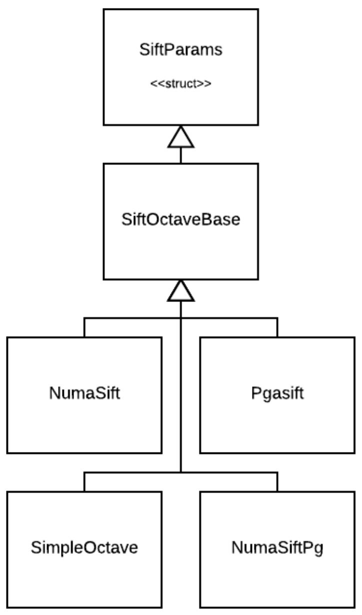


Figure 3: Class hierarchy of the various *SIFT* implementations

3.2 Numasift data flow

The implementations of `SimpleOctave`, `NumaOctave`, and `NumaOctavePg` have an important detail in common. While they implement parallelism, they implement it *per octave*. That means that an octave object will spawn multiple threads, and once an octave has been completed, processing will move on the the next octave. Thus, all data that is used will be present in the main memory of a single node, as illustrated in figure 4.

There is also good reason to order processing in this fashion. Every octave takes a pre-processed image from the previous octave as an input. This pre-processed image has several Gaussian filters applied and has been halved in size. By cascading these blurring-and-halving-steps, a considerable amount of computational effort is being avoided in comparison to doing it over and over again for every octave.

3.3 Adapting Numasift for PGASUS

We believe that the central storage of all image data on a single node accessed by many nodes leads to numerous avoidable remote memory accesses. Furthermore we believe that every octave should do processing in parallel with other octaves, with every octave being mapped on to a specific node. Figure 5 illustrates the envisioned co-location of work and data per node and octave.

As mentioned in the previous subsection, parallel octave processing requires every octave to do input image processing on their own. This required a different approach with `PgasusOctave` in comparison to `NumaOctave`, although a lot of the

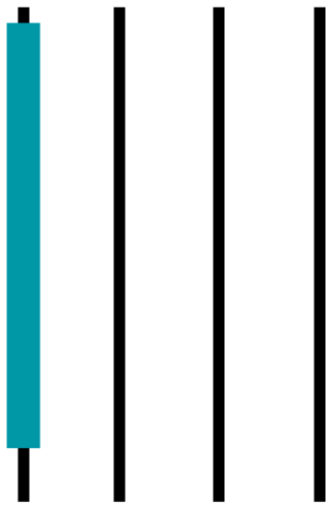


Figure 4: Sequential processing of a sequential node with all processed data (blue) lying central on a single node

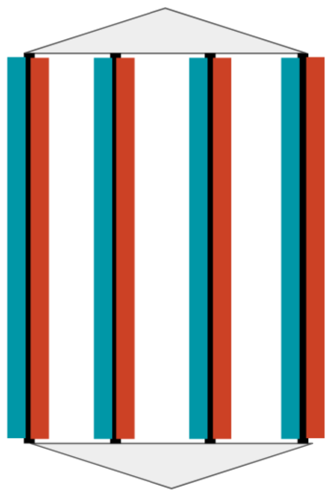


Figure 5: Parallel processing with data (blue) and work (red) happening on every node. The triangles indicate the initial distribution of work and final collection of results

former is based on the latter. We did this to maintain comparability with the testing methodology from Plauth et al. [3].

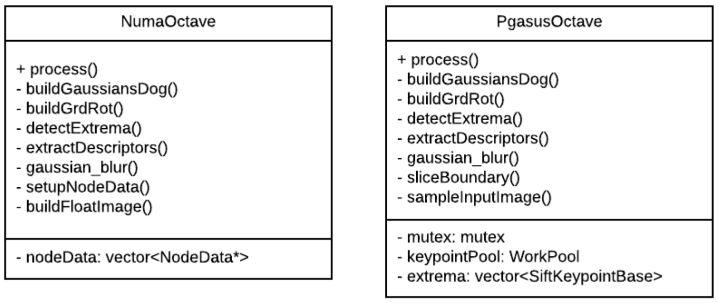


Figure 6: Differences in class methods of PgasusOctave and NumaOctave

As figure 6 shows, the `nodeData` array carrying the node-local temporary constructs has disappeared since all processing now takes place on a single node. With the aforementioned difference in execution order, a method `sampleInputImage` to sample and blur the image to the right size and degree has been introduced. It is called in the constructor.

All logic involving node-specific memory placement takes place in the function `sift_pgasus`, that is also implemented in `pgasus_octave.cpp` and thus the class does not need to have and thus the class does not need to have any knowledge of its MemorySource.

4 Outlook

Our goal for the upcoming *Spring 2018* period of the Future SOC Lab is to shift our focus towards high-level integration mechanisms for hardware accelerators, most prominently GPUs and possibly even FPGAs. In addition to evaluating high-level APIs for accelerator utilization, we are also planning to evaluate additional techniques such as transparent compression for improved resource utilization.

References

[1] W. Hagen, M. Plauth, F. Eberhardt, F. Feinbube, and A. Polze. “Pgasus: A framework for c++ application development on numa architectures”. In: *2016 Fourth International Symposium on Computing and Networking (CANDAR)*. Nov. 2016, pages 368–374. DOI: 10.1109/CANDAR.2016.0071.

[2] D. G. Lowe. “Object recognition from local scaleinvariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Volume 2. 1999, pages 1150–1157.

[3] M. Plauth, W. Hagen, F. Feinbube, F. Eberhardt, L. Feinbube, and A. Polze. “Parallel implementation strategies for hierarchical non-uniform memory access systems by example of the scale-invariant feature transform algorithm”. In: *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. May 2016, pages 1351–1359. DOI: 10.1109/IPDPSW.2016.47.

Machine Learning for self-serviced Electrocardiograms

Tim Cech, Noel Danz, Nele Noack, Lukas Pirl, Jossekin Beilharz, and Andreas Polze

Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.uni-potsdam.de

Electrocardiograms today are manually interpreted and used as a basis for diagnosis. The use of electrocardiographs that can be operated by the patient himself open new possibilities of diagnosing heart arrhythmia and other conditions. At the same time they increase the amount of data that needs to be processed. While the final diagnosis will remain the task of a doctor for the near future, there is already now potential for an automated system to prioritize electrocardiograms. This report presents the work on automatic classification of heartbeats on modern hardware architectures. We show that an accuracy of 98.5 % is possible with different types of neural networks.

1 Introduction

As more and more data is recorded the need for efficient data prediction grows. The medical world is no exception to this phenomenon. In our work we focused on analyzing heartbeats taken from raw ECG data. For this we partnered up with *GETEMED Medizin- und Informationstechnik AG*, a medium-sized enterprise located in Teltow, who mainly produce hardware and software for the analysis of ECG data.

The automatic classification of heartbeats into different categories like normal beats, wide beats or artifacts greatly helps doctors in assessing an Electrocardiogram and diagnosing a heart condition, as shown in figure 1.

Our work with the Future SOC Lab resources investigates the use of modern hardware architectures for the automatic classification.

2 Background

There are different ways to classify the heartbeats from ECG data sets. Currently, the industry partner in this project, the GETEMED AG, uses a manually engineered algorithm that classifies heartbeats. It does so by checking for predefined thresholds, e.g. the length of the beat and the amplitude a certain period of time after the peak. Contrastingly, this work evaluates approaches to classify heartbeats from ECG data sets with machine learning techniques.

Specifically, we employ artificial neuronal networks for the classification. The idea behind neuronal networks is to imitate the human brain — of course, in a drastically simplified manner — in order to enable computers to "learn" to fulfill certain tasks.

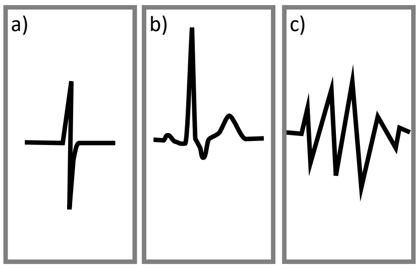


Figure 1: Heartbeats that have been automatically recognized should be classified into normal beats (a), wide beats — like the right bundle branch block shown in (b), or artifacts that were recognized as beats (c).

For example, such a task could be to decide if a cat is contained in an image or not. An artificial neuronal network consists of vertices (*neurons*) connected through weighted edges (*synapses*). In the initial training phase of the neuronal network, the weights of the edges are being adjusted to maximize the network's performance regarding the specific task. For this training, a data set for which the desired results are known a priori must be used. Finally, the neuronal network fulfills its task based on the trained edges weights.

For our training and testing we mainly used the *Physionet MIT-BIH arrhythmia database* [4, 7]. This database is considered the gold standard of annotated ECG data. It is reviewed by two independent physicians and is therefore considered a class 1 database [4].

Table 1: Precision, Recall, F1-score are the values in respect to the class of normal beats

Type	Accuracy	Precision	Recall	F1-score
MLP	98.45 %	98.54 %	98.33 %	98.44 %
DTW DT	65 %	73 %	68.4 %	70.62 %
LSTMN	98.87 %	98.86 %	98.90 %	98.87 %

For our use case the data source has such a high quality that we are currently missing enough artifacts to enable a three way classification into normal, wide heartbeats as well as artifacts. A QRS complex "too noisy to be correctly classified" is called an artifact. In addition, there are noisy sections that look so similar to a QRS complex that they are triggered as such and must therefore also be recognized as artifacts. The review of two independent physicians is not practical in real world scenarios, which makes increasing the amount of training data difficult.

3 Current Status

In the last six month we investigated the use of different types of neural networks, in multiple areas of interest. We developed a prototype for Generative Adversarial Networks, Dynamic Time Warping (DTW), Multilayer Perceptrons (MLPs) and Long short-term Memory Networks (LSTMNs).

In the area of neural networks, we leverage two concrete architectures: Deep Neuronal Network (DNN) and LSTMN. DNNs refers to the concept of having multiple layers of neurons. The concept of LSTMNs introduces special neurons, which feature more sophisticated input/output logic, as well as memory to enable stateful behavior.

For the pre-processing of the data, we take advantage of DTW and Decision Tree Classification (DT) technologies. Both serve as a filter to separate out ECG data which is unusable for the analysis (e.g. due to improper placement of measurement electrodes). While DTW is an approach to quantify the similarity of sample sequences, DTs are an abstraction to represent decision algorithms (e.g. classification).

We also covered the theoretical work for a Reinforcement Learning (RL) solution and a Convolutional Neural Network (CNN), but these approaches where not tested in practice so far.

3.1 Approach

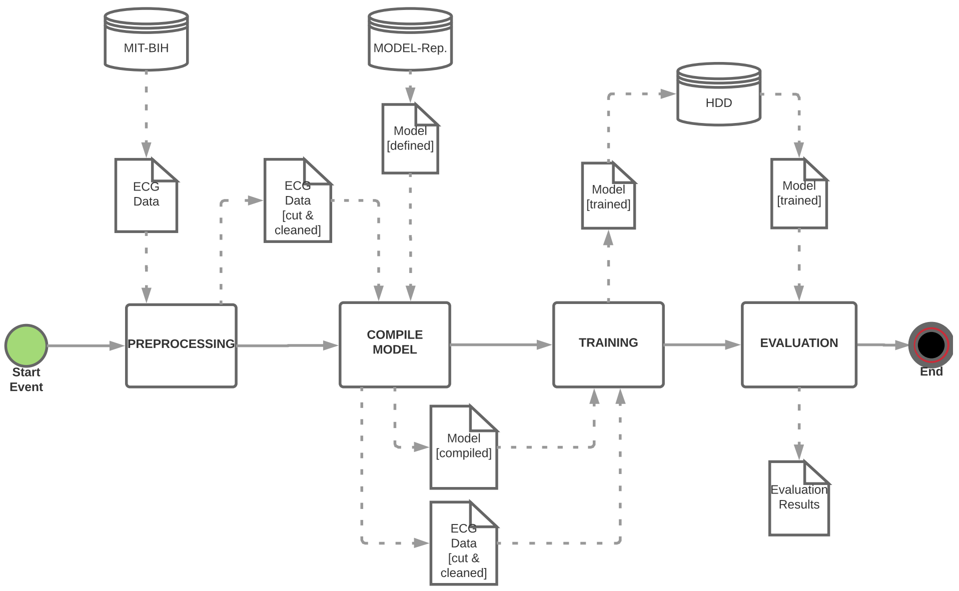


Figure 2: The pipeline used to classify heartbeats

The work presented uses Tensorflow [1], Keras [3] and scikit-learn [8] to build and test the machine learning models. The training data exists in the WFDB data format. It is accessed and displayed by using the WFDB package for python [11]. After loading of the data from the database the electrocardiograms are then cut into single heartbeats using the annotations for beats present in the dataset. After splitting the data into sets for training and testing, the respective models are trained and evaluated. Inside this overall approach the following techniques were evaluated:

3.1.1 Generative Adversarial Network (GAN)

The GAN consists of two networks, the generator and the discriminator. The generator gets a vector of random data and generates a realistic looking ECG. The discriminator learns to distinguish between the ECG data generated by the the generator and the actual training data from BIH-MIT. The generator consists of a MLP with two hidden layers it scales the 100 dimensional vector to the needed number of samples. The discriminator is structured the same way but reverses the process and reduces the ECG beat to a single number describing if the ECG beat is part of the real dataset.

3.1.2 Dynamic Time Warping (DTW)

DTW [2] is an algorithm to pre-process time-continuous data. First we cut the beats out of the record with the wfdb-functions, after that we preprocess the data of record 200 with DTW. We used the first beat of record 100 as the template curve. Then we train a DT over the results. Afterwards we let the DT classify our test data. Our test data is preprocessed with the same DTW-algorithm.

3.1.3 Multilayer Perceptron (MLP)

We developed several MLPs to classify our data. Our best MLP had two hidden layers with 64-32-16 neuron architecture. This approach especially fits in our use-case, because we want to let the network learn the abstract concept of an artifact or rather of a wide heartbeat. This requirement of trying to learn an abstract concept is often well addressed by MLPs. [9]

3.1.4 Long short-term Memory Network (LSTMN)

We developed a 2-stack LSTMN [5] with a two hidden layer MLP as the summarizing layers to classify our dataset.

3.1.5 Reinforcement Learning (RL)

We also looked into the usability of RL techniques [10] to classify our dataset. Unfortunately the typical use-case for RL is the interaction with an environment. As we do not have an interactive environment, we are missing the means to define a reward system. A typical use case would be learning how to play a game as in [6].

3.2 Results

The results of our experiments are shown in table 1. Only the best results with the same technique are shown. The results of the experiments with DTW are quite bad compared to the results of the experiments with MLP or LSTMN. Both techniques scored about 98 % in all measures. So we considering these techniques really promising and superior to DTWalgorithms.

In general the training phase of LSTMNs is really slow compared to MLPs. On the one hand we could train a MLP approximately 500 epochs and in the same time a LSTMN only 10 epochs, on the other hand a LSTMN score about equal after 10 epochs compared to a MLP after 500 epochs.

4 Future Work

In the future we would like continue to improve the models to classify heartbeats in ECG data.

For this we want to train on lager datasets that especially also allow to learn the recognition of artifacts, as artifacts occur often in practice, but are almost absent from the Physionet MIT-BIH arrhythmia database. Furthermore we want to integrate k-fold cross validation to get more general findings and evaluate the use of feature extraction as further input to the network.

5 Conclusion

In this work, we evaluated the use of modern hardware architectures for the automatic classification of heartbeats in electrocardiograms. We showed MLP and LSTMN to perform very well for our datasets and are looking forward to conclude this work in the future.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems*. White Paper. 2015. arXiv: 1603.04467 [cs.DC]. URL: http://download.tensorflow.org/paper/whitepaper2015.pdf.

[2] D. J. Berndt and J. Clifford. "Using dynamic time warping to find patterns in time series". In: *KDD workshop*. Volume 10. 1994, pages 359–370.

[3] F. Chollet and Keras contributors. *Keras*. 2015. URL: https://github.com/keras-team/keras.

[4] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. *PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals*. Circulation, 101(23):e215–e220, 2000. DOI: 10.1161/01.CIR.101.23.e215.

[5] A. Graves and J. Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5 (2005). IJCNN 2005, pages 602–610. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2005.06.042.

[6] S. Karakovskiy and J. Togelius. "The mario ai benchmark and competitions". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012), pages 55–67. DOI: 10.1109/TCIAIG.2012.2188528.

[7] G. B. Moody and R. G. Mark. "The impact of the mit-bih arrhythmia database". In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pages 45–50. DOI: 10.1109/51.932724.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (Nov. 2011), pages 2825–2830. ISSN: 1532-4435.

[9] Y. Qian, Y. Fan, W. Hu, and F. K. Soong. "On the training aspects of deep neural network (dnn) for parametric tts synthesis". In: *Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pages 3829–3833. DOI: 10.1109/ICASSP.2014.6854318.

[10] R. S. Sutton and A. G. Barto. "Reinforcement Learning: An Introduction". In: 9.5 (1998), pages 1054–1054. DOI: 10.1109/TNN.1998.712192.

[11] C. Xie, L. McCullum, A. Johnson, T. Pollard, B. Gow, and B. Moody. *Waveform Database Software Package (WFDB) for Python*. 2021. DOI: 10.13026/G35G-C061. URL: https://physionet.org/content/wfdb-python/3.3.0/.

Modelling of aluminum reduction processes with machine learning approaches and mathematical models using SAP HANA in-memory computing for smart-devices

Roberto C. L. De Oliveira and Fábio M. Soares

Post-Graduation Program in Electrical Engineering
Federal University of Pará, Brazil
limao@ufpa.br, fms@ufpa.br

Metallurgical modelling based on the physical laws (white box) purely covers all the theoretical phenomena in the plant, but its development takes too much effort, and moreover the model should be tuned against real data, which are often noisy. On the other hand, a purely data driven modelling (black box) requires less effort and provides a simpler model, but includes a significant bias as the plant history is the only foundation of this kind of model. Since a good process model should be prepared to reproduce any behavior a plant might present, even if it never happened, we thought of a hybrid modelling (grey box) considering the response of theoretical models along with the actual data to find simple process models that can be realized in small smart-devices. However, a lot of processing and exhaustive optimization is required in these tasks, making mandatory the use of parallel computing and in-memory facilities. The result of this work is to develop an approach to model a complex problem like Aluminum Reduction using the physical already known models as data generators to add further data into the existing dataset, which should be used a source for machine learning algorithms.

1 Project Idea

The metallurgy industry involves several disciplines and competencies. In addition, production cells are often subject to disturbances and unexpected deviations, making the process control an even harder challenge [4]. Process models help in forecasting variables, giving trends, simulation of what-if scenarios and tests as well [1, 5, 7]. An accurate model depends on decent quality data covering a range of process operations as wide as possible. However, data-driven modeling (black box) often hinders this range, because plants usually work under closed loop, therefore representing only the plant's steady state [2, 3, 6]. On the other hand, a physical modelling (white box) is too theoretical, sometimes involving nearly unsolvable equations or calculations that would take a lot of effort, and often the models need a fine-tune to adjust to real data [6]. Keeping both paradigms in mind, it is encouraged the use of both approaches in building a stronger model combining their good characteristics together. The white box models can be used to produce complementary data to help in the building of a black box model. This fact encourages use of high performance computing resources, like parallelism and in-memory computing. What motivated this work was the possibility of using the FSOC powerful hardware to perform massive calculations jointly with the SAP HANA® in-memory capability.

Aluminum is produced by the Hall-Héroult process, an electrolysis based production by which aluminum is extracted from the alumina molecules under a strong electric current [1, 4, 5, 7]. This strong current causes the heating of the electrolyte, also known as bath, which is a mixture of several chemical species. This chemical composition causes variations in the bath temperature as well as in the physical state of the bath. Figure 1 shows a schema of a typical production cell.

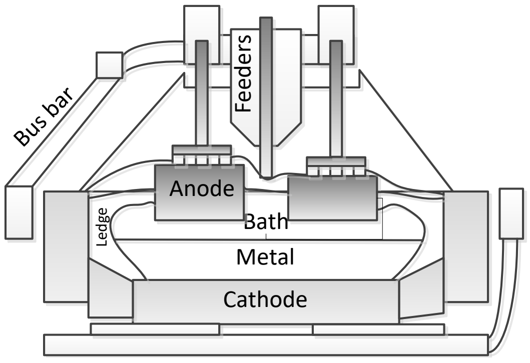


Figure 1: A typical aluminum reduction cell

Aluminum is produced by electrolysis through the passage of a 100 kA to 500 kA current from anode (+) to cathode (-) allowing the production reaction:

$$2Al_2O_3 + 3C^+ \rightarrow 4Al^+ + 3CO_2$$

However, an excess of alumina concentration and bad dissolution causes the back reaction, which turns aluminum back into alumina, therefore reducing efficiency:

$$2Al^+ + 3CO_2 \rightarrow Al_2O_3 + 3CO$$

For a maximum efficiency, important variables like Cell Resistance, Cell Temperature, Metal purity and Chemical Concentration of species, among others, are monitored. Others, like Decomposition Voltage and Solubility are not directly measured but can be inferred though models. For each of these variables there is a set point range, however under certain conditions this setpoint may be flexible, according to the cell status.

Aluminum smelters usually have a large number of cells, which in their turn are divided into sections, rooms and lines (also known as potlines or reductions) [4]. Provided that many phenomena are not yet fully understood by smelters, not every cell present the same behavior. In this sense every cell needs a specific control considering their peculiarities as well as their own parameters. Control systems collect a lot of data from a lot of cells, making the process database very huge, nevertheless these measurements are still just a small part of what happens in the process, since many variables (anode-to-cathode distance, alumina consumption and ledge composition) cannot be measured. On the other hand, with so many modeling works available in the literature, these variables can be estimated. In addition, not every variable can be measured every time, and some variables are measured only on-demand.

2 Used Future SOC Lab Resources

The concerned approach consists of two steps. First is to expand the original database to include other process variables and the estimates when they were not measured.
The concerned approach consists of two steps. First is to expand the original database to include other process variables and the estimates when they were not measured.

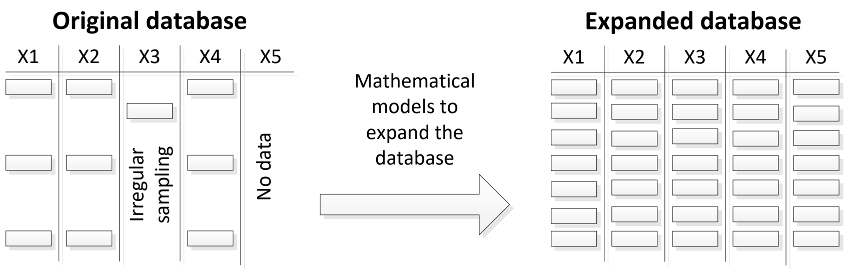


Figure 2: Schema of database expansion

In figure 2, five variables (X1 to X5) are shown, from which only X1, X2 and X4 are directly measured, X3 has irregular sampling and X5 has no measurements. We use theoretical mathematical models to generate data not present in the original database. At the end, all the data from the concerned variables will be present in the database.

The step 2 is to use machine learning techniques on the full process database. In this phase the model to be built should cover a wider operating range, that means the data should be selected considering all the operating range, however outliers should be dropped. Subsequently, using in-memory and parallel processing of SAP HANA Predictive Analytics plugin we exhaustively train several supervised machine learning algorithms that are suitable for executing in cheaper hardware such as IIoT smart devices.

For this project we use four large (L) and four extra-large (XL) virtual machines of the Future SOC Lab to run the mathematical models written in Python 3.5 to generate all the data needed, two (L and XL) for each potline (our database contains data from 4 potlines). The data are saved in a SAP HANA instance where the machine learning models are built upon the data using the in-memory processing. In the previous version of this work, a neural network with the SAP HANA AFL was created for

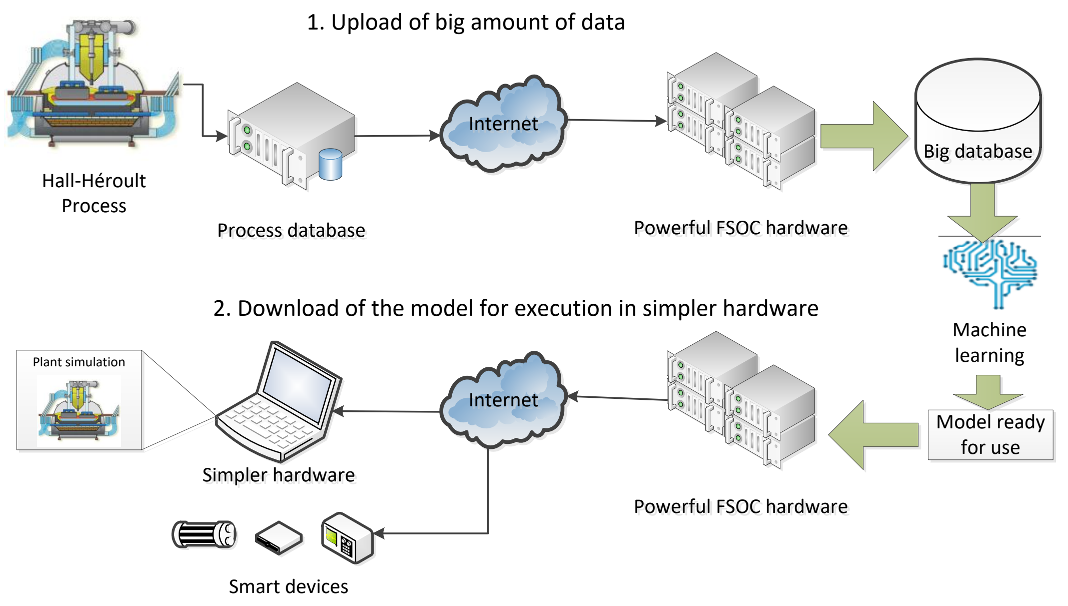


Figure 3: Overall process for building the models, from uploading the data (upper line # 1), building the model using machine learning approaches, to downloading the final model for use in laptops and smart devices

4 variables. To model other variables we run other supervised learning regression algorithms. To validate the models, we applied the analysis on the error/residue. Additionally, we simulated some situations that happen in the real process to verify how the model reacts.

The primary database comprised about 100 concerned variables from about 900 cells over a period of six years and 24-hour sampling was to be uploaded into the SAP HANA tenant. From these, 22 variables have been uploaded. An additional (secondary) database was provided by the smelter containing online data from the control systems and 0.5-second sampling, however only the data was collected from only 33 cells. The secondary database contains 9 variables and was used to generate additional (calculated and modelled) variables. We applied three mathematical models, either to resample the database to 4 hours and include variables whose measurement are not possible, such as solubility and enthalpy.

3 Results of this approach

In the previous research period we found the optimal configuration for multilayer perceptrons (MLP) with one single hidden layer to model four of the variables (Bath Temperature, Aluminum Fluoride, Resistance and Alumina), whose inputs consisted of 45 variables, from these 15 are original from the database, 10 are external control and 20 have been generated by mathematical models. Except for alumina, the MLP neural networks successfully mapped the input output behavior, with an average normalized mean squared error (MSE) of 1.5×10^{-4}. The MLP neural networks were trained by varying the hidden layer size from 1 to 100 neurons and used up to three lags for each input variable. The training dataset comprised 85 % of all data and the remaining 15 % was used for testing.

We kept these neural network configurations and modeled in this new period another 4 variables (Calcium Fluoride, Liquidus Temperature, Metal Level and Decomposition Voltage) using the same approach. In addition, we applied Principal Components Analysis (PCA) to eliminate possible redundancies in the input data, in order to work only with rich-information data, and we performed simulations by feeding the neural network's output back into the input. In figure 4 are the scatter charts for each of the new variable modeled during training.

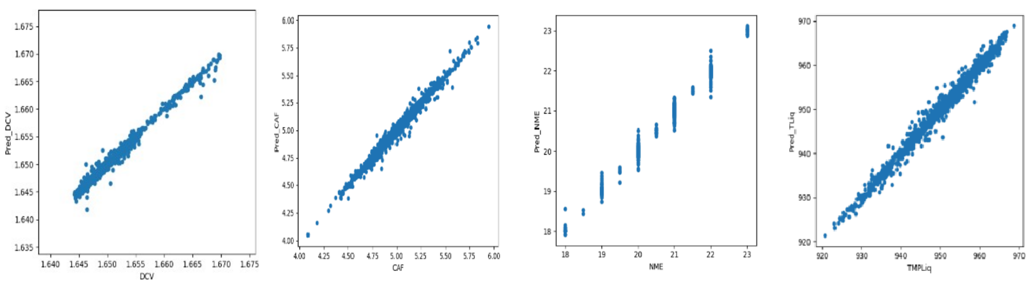


Figure 4: Scatter plots of four process variables, Decomposition Voltage (DCV), Calcium Fluoride (CAF), Metal Level (NME) and Liquidus Temperature (TMPLiq) modeled in this period

The decomposition voltage is calculated by a mathematical model, i.e. the accuracy of the neural network is dependent on the accuracy of the theoretical model, whose uncertainties might be inputted in the neural network as a noise, therefore producing outliers in the predictions. The other three variables are present in the original database, the distribution of the prediction vs real values follows an acceptable pattern. The case for Metal Level is distinct because the values are measured in the process using a discrete scale, but the neural network produces real values in a continuous scale. One additional remark is for the Liquidus Temperature, whose value is strongly affected by chemical composition of the bath and ledge. Since we cannot measure the ledge variables, but they can be estimated by models, there is always an error in the liquidus temperature of about ±3 °C.

The chart in figure 5 shows the training and test error according to the number of hidden neurons for each variable neural model.

With the best neural networks obtained we simulated an aluminum production cell working on feedback, i.e. the neural network output is fed back into the input.

Table 1 below shows the best neural network so far found for all the eight variables modeled.

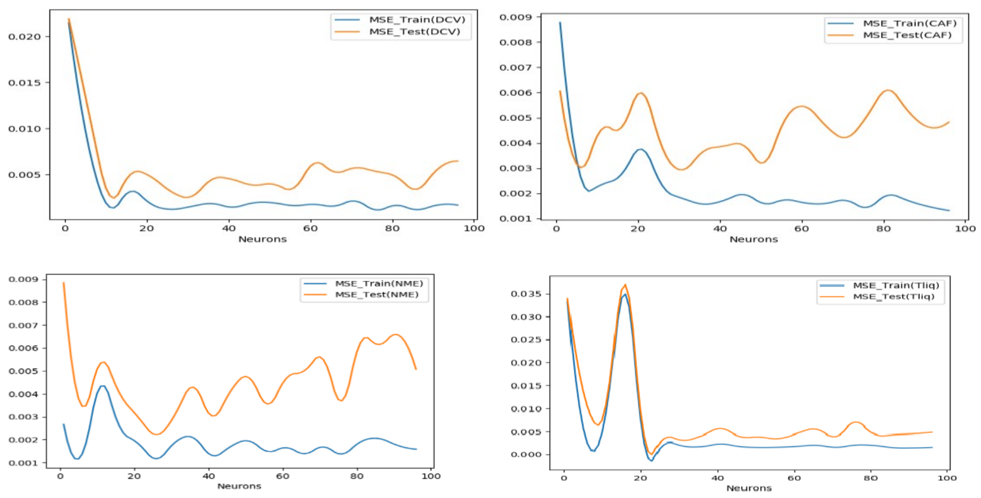


Figure 5: Training and Test MSE vs. the number of hidden neurons for each variable

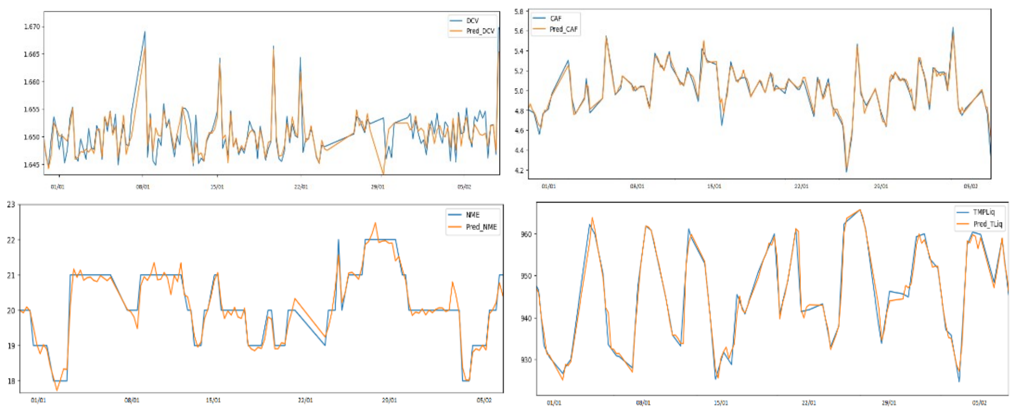


Figure 6: Simulation on a given cell model for each of the variables Decomposition Voltage(DCV), Calcium Fluoride (CAF), Metal Level (NME) and Liquidus Temperature (TMPLiq)

Table 1: Neural networks so far

Variable	Neurons	Training MSE	Testing MSE	Prediction Correlation
Bath Temperature (TMP)	66	0.000 78	0.001 083	0.9962
Aluminum Fluoride (ALF)	61	0.000 668	0.001 28	0.9918
Cell Resistance (RMR)	42	0.000 961	0.001 815	0.9945
Alumina (ALU)	81	0.002 315	0.004 563	0.9277
Decomposition Voltage (DCV)	36	0.001 539	0.002 624	0.8975
Calcium Fluoride (CAF)	31	0.001 78	0.002 932	0.9635
Metal Level (NME)	27	0.001 168	0.002 216	0.9744
Liquidus Temperature (TLiq)	30	0.001 842	0.003 025	0.98

4 Future steps

In this phase of the project we performed PCA to reduce dimensionality and simulated the models by feeding back its predictions, so we could compare how reliable they are in the long run. The reliability of the mathematical models also requires attention, as new studies on the electrolyte might reflect on new parameters for the models we applied here to generate the new data. We also should consider using spatial data from the solutions of partial differential equations of the models.

References

[1] P. Biedler. *Modeling of an Aluminum Reduction Cell for the Development of a State Estimator*. Dissertation submitted to the College of Engineering and Mineral Resources at West Virginia University, 2003.

[2] U. Forsell and L. Ljung. "Closed-loop identification revisited". In: *Journal Automatica IFAC* 35 (1999), pages 1215–1241. DOI: 10.1016/S0005-1098(99)00022-9.

[3] L. Fortuna, S. Graziani, A. Rizzo, and M. Xibilia. *soft Sensors for Monitoring and Control of Industrial Processes*. London, UK: Aluminium Verlag, 2007. ISBN: 978-1-84628-480-9.

[4] K. Grjotheim and H. Kvande. *Introduction to Aluminium Electrolysis*. Düsseldorf, Germany: Aluminium Verlag, 1993. ISBN: 978-3-410-22027-5.

[5] S. W. Jessen. *Mathematical Modeling of a Hall-Héroult Reduction Cell*. Dissertation in partial fulfillment for the M. Sc, 2008.

[6] L. Ljung. *System Identification: Theory for the User*. 2nd edition. USA: Prentice Hall PTR, 1999. ISBN: 0-13-656695-2.

[7] A. W. Wright. "The dynamic simulation and control of aluminium smelting cells". PhD thesis. Newcastle University, School of Chemical Engineering and Advanced Materials, 1993. HDL: 10443/1560.

Energy Efficiency, Virtualization and Performance

Carlos Juiz and Belen Bermejo

Computer Science Department
University of the Balearic Islands, Spain
{cjuiz,belen.bermejo}@uib.es

1 Project idea

The prevailing international scientific opinion on climate change is that human activities resulted in substantial global warning from the mid-20th century. That continued growth in greenhouse gas concentration, caused by human-induced emission, is mainly (direct or indirectly) due to energy consumption. The sector where energy consumption has grown tremendously is IT (Information Technology). On one hand, data centers that host cloud services is becoming huge warehouses with a lot of servers, additional electronic equipment and complex cooling systems. These computers and telecommunications networks are today primarily responsible for electrical energy consumption caused by data centers, which maintain the quality of internet services in operation. Giving more capacity to these data centres usually means more virtualized servers and consequently additional more equipment and cooling are needed, too. On the other hand, the increasing number of users, especially due to popularization of cloud services, produces continuously capacity problems at data centres due to performance requirements [3]. Thus, a new challenge directly related to this area of research emerges: the energy efficiency of cloud systems [2]. Additionally, new server virtualization technologies are appearing and have significant benefits in terms of hardware utilization, energy-efficiency and configuration flexibility for its performance. This project is aimed by this combination of these topics. The main objectives are the following:

1. To Characterize server energy consumption through monitoring, benchmarking and modeling.

2. Modeling energy consumption patterns and performance of server virtualization. We shall infer models of such energy, performance and virtualization, together.

3. Performing several comparative studies of benchmarking between physical and virtual servers. Virtualization can be used, among other benefits, to reduce the number of active servers and thus to reduce energy consumption. However, the prices to be paid surely are: performance overhead, probability of additional failures due to virtualization software and less longevity of components due to hardware overutilization. The main goal of the proposed project is to research how server virtualization positively affects to the overall energy consumption

but negatively in other factors due to the additional layers of software. **This subtle trade-off is the primary concern of this research project.**

4. To validate our performance-oriented previous findings and energy efficiency virtualization models through experimentation at lab and real data centres.

From the aims defined before, in this stage we achieve the first one, that is, the characterization of server energy consumption and performance. We started to study the performance and energy of virtualized servers under a CPU-intensive workload.

Then, the first research question we attempt to answer is: "How to measure the goodness of virtual machine consolidation taking into account the performance-energy tradeoff"?

2 Used Future SOC Lab resources

In order to answer the research question, we design a methodology based in the following stages:

- Characterize the performance and energy consumption of physical servers through benchmarking and monitoring techniques [4].
- Quantify and describe the performance-energy trade-off through a proposed index [1]. This index, namely CiS^2 attempts to quantity the trade-off between the performance and energy consumption in consolidated servers.
- Measuring the proposed index in different physical machines.
- Discussing and to analyse the results.

The exposed methodology was performed in the HPI Future SOC Lab infrastructure, specifically in the rx600s5-1 server which hardware has 48 CPUs and 1024 GB of RAM memory. In Table 1 we can see the actions developed in each stage and the correspondent outcome.

3 Findings

In previous section we explain the stages we perform to answer the research question, as well as the developed actions and its outcomes. In Fig. 1 the graphical representation of the CiS^2 comparison between our physical servers and the HPI infrastructure. We can observe that the HPI infrastructure has the best CiS^2 value for any number of consolidated virtual machines.

Then, if the physical server has better hardware, it is able to manage the virtual machines in a better manner. As a consequence, the performance and energy effects of the virtual machine consolidation are lesser than in other physical servers, as our lab.

Table 1: Actions developed in each stage and its outcomes

Stage	Action	Outcome
Stage 1	Benchmarking and monitoring the HPI server under a CPU-intensive workload	Performance and power consumption data
Stage 2	Calculation of CiS^2 index	Behavior of CiS^2 index in HPI server
Stage 3	Representation of CiS^2 in HPI and other physical servers from our lab	Graphical representation
Stage 4	Analysis and comparison of HPI's CiS^2 and our lab	Graphical representation and comparison

3.1 Publications

The use of the HPI infrastructure collaborates to develop the following research works:

- B. Bermejo, C. Juiz, and C Guerrero. "On the Linearity of Performance and Energy at Virtual Machine Consolidation: the CiS2 Index for CPU Workload in Server Saturation". In: Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications. Exeter, UK, 2018, pages 928– 933.

- B. Bermejo, C. Juiz, and C. Guerrero. "Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance". In: The Journal of Supercomputing (2018). issn: 1573-0484. doi: 10.1007/s 11227-018-2613-1.

4 Next steps

In this project we attempt to answer the research question: "How to measure the goodness of virtual machine consolidation taking into account the performance-energy tradeoff?". For that, we design a methodology based on monitoring and benchmarking techniques and we apply them to our servers and then, we extend the experiment to HPI infrastructure.

The HPI allows us to extend the work and also to obtain more accurate results and conclusions. Due to that, there are very interesting research questions that we would like to answer with the support to the HPI infrastructure. The questions are the following:

- How the behavior of the CiS^2 index under different workload is?

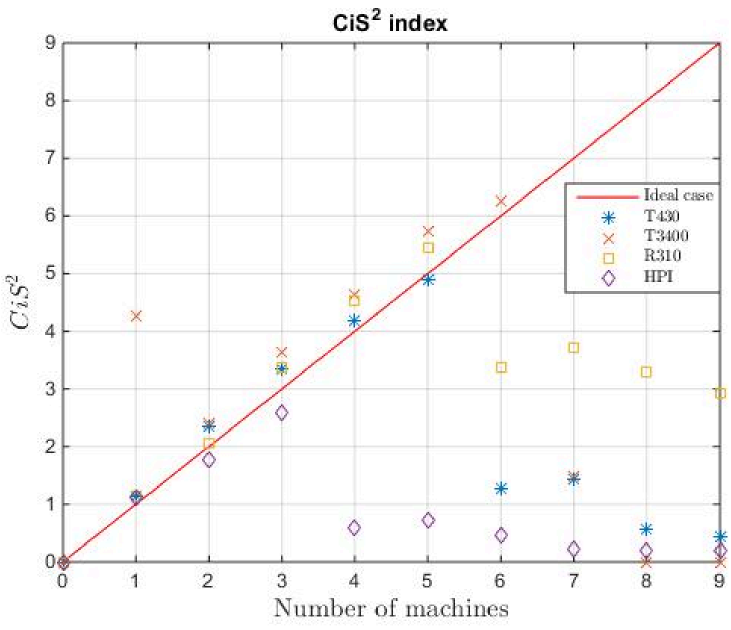


Figure 1: CiS^2 index comparison

- Could be generalize the behavior of the CiS^2 index?

In order to answer the proposed questions, we will apply for a new project in the Hasso Plattner Institute in order to use the HPI Future SOC Lab's IT infrastructure.

References

[1] B. Bermejo, C. Juiz, and C. Guerrero. "On the Linearity of Performance and Energy at Virtual Machine Consolidation: the CiS2 Index for CPU Workload in Server Saturation". In: *Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications*. Exeter, UK, 2018, pages 928–933.

[2] B. Bermejo, S. Filiposka, C. Juiz, B. Gómez, and C. Guerrero. "Improving the Energy Efficiency in Cloud Computing Data Centres Through Resource Allocation Techniques". In: *Research Advances in Cloud Computing*. Edited by S. Chaudhary, G. Somani, and R. Buyya. Singapore: Springer Singapore, 2017, pages 211–236. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_9.

[3] R. Buyya, C. Vecchiola, and S. T. Selvi. *Mastering Cloud Computing: Foundations and Applications Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 978-0-12-411454-8.

[4] X. Molero, C. Juiz, and M. Rodeño. *Evaluación y Modelado del Rendimiento del os Sistemas Informáticos*. Pearson, 2004. DOI: 10.2200/S00516ED2V01Y201306CAC024.

BPChain: A Benchmarking Tool for Comparative Evaluation of Blockchain Protocols

Jonas Cremerius, Simon Siegert, Anton von Weltzien, Annika Baldi, Finn Klessascheck, Svitlana Kalancha, and Tom Lichtenstein

Hasso Plattner Institute, University of Potsdam, Germany
{first.last}@student.hpi.uni-potsdam.de

Blockchain technology, promising to transform traditional business networks, quickly attracted attention of enterprises. In response, many blockchain protocols were released, offering different benefits for business. Thus, it significantly complicated the choice of the technology for blockchain powered applications . However, so far no universal comparison framework, which provides objective measures of different protocols, exists. Aiming to perform a comparative evaluation of blockchain protocols we are working on a benchmarking tool, which will give users the possibility to choose the best fit for a concrete scenario. The platform provides a real-time visualisation of metrics describing the blockchains' performance. Furthermore, the platform allows to run scenarios on configured blockchains, which helps users to choose the right blockchain protocol for a specific case. This paper presents our results, which includes the approach, the architecture, and first benchmarking results.

1 Introduction and Motivation

Blockchain technology — based on distributed ledger technology — allows to optimise business operations and make them more transparent, thus, enabling trustless interactions between participants. Many applications using this evolving technology are now available, and in response, several blockchain platforms have been developed. These platforms have different characteristics and offer various capabilities empowering users to create their own blockchain applications. The challenge is to choose the right platform for the specific use case. A benchmarking tool should meet following requirements to provide a qualitative analysis:

- Provide support for different blockchain technologies
- Identify specific and independent metrics for evaluation of performance of blockchain protocols
- Allow configuration and setup of blockchain parameters
- Run different scenarios on blockchain networks to compare their performance for a concrete use case.

In general, blockchain platforms can be classified into two main types—permissionless and permissioned. Permissionless systems such as Ethereum [5] or Bitcoin [3] are publicly available for use. Even if the network is configured as private (in the case of Ethereum), each participating node still can conduct transactions as well as take part in the consensus process to run the blockchain. Permissioned platforms, such as MultiChain or Hyperledger, allow to build a so called consortium, where every new node must be approved before joining the network and conducting transactions. To discover strengths and weaknesses of both blockchain network types, we deployed and tested Ethereum, XAIN, and MultiChain systems.

- Ethereum is a general purpose blockchain platform that supports smart contracts and is open and permissionless. It presently uses its own Proof of Work (PoW) consensus protocol [5].
- MultiChain is a permissioned platform for creation and deployment of private blockchains, either within or between organizations. MultiChain utilizes the Byzantine Fault Tolerance (BFT) consensus protocol and does not include support of smart contracts [2].
- XAIN (eXpandable Artificial Intelligence Network) is a framework for the development of intelligent and adaptive blockchain platforms. The XAIN framework supports a new consensus protocol, Practical Proof of Kernel Work (PPoKW), a variant of Proof of Work based on Cryptographic Sortition. Furthermore, XAIN employs various techniques aimed at improving all measurable runtime properties. The XAIN blockchain supports smart contracts [9].

2 Related Work

The latency of blockchain based systems has already been investigated by Yasaweerasinghelage et al. [10]. In the paper they investigated the feasibility of using architectural performance modelling and simulation tools to predict the latency of blockchain-based systems. In another paper, Weber researched the question of read and write availability in transaction management, comparing Ethereum and Bitcoin networks [8]. Furthermore we looked into the project of National Singapore University, which aims to create a framework for analysing private blockchains. The project focused on evaluation of three private blockchains: Ethereum, Parity, and Hyperledger Fabric. The research concluded that these systems are still far from replacing current database systems due to gaps in performance [1]. The review of related work shows that the evaluation framework for blockchains is researched actively and that further evolvement of the technology plays a big role for enterprises.

3 BPChain: A Solution for Blockchain Benchmarking

3.1 Estimation Criteria and Blockchain Metrics

Each tested technology has strengths and weaknesses, depending on the underlying consensus algorithm and network configuration. The choice of platform is determined by the overall efficiency, scalability, security and maturity of its blockchain, which can be evaluated through the network parameters.

Scalability of the blockchain network is its ability to reach consensus when the number of peering nodes are constantly increasing. Networks using PoW have significant problems with scalability: more peers in the network and therefore more transactions result in longer time to reach consensus and to create a block. If the block size parameter is set too low, more blocks will be needed to cover transactions resulting in longer consensus time [7]. Therefore, generally an Ethereum chain is difficult to scale, while MultiChain does not experience scalability problems.

Efficiency of a blockchain is measured with two criteria: speed and cost of transactions. Ethereum demands a certain fee to conduct a transaction, whose size grows with the number of nodes in the network. The geography of miners as well as their computing power define the throughput limit of the blockchain, thus the speed of transactions' confirmation. Blockchain protocols operating PoW have a lower transaction throughput, because each block of valid transactions must be verified and stored in the chain, demanding high computation power. BFT-based blockchains, such as MultiChain, operate without mining and transaction fees ensuring higher transaction throughput.

Security of a blockchain relies on robustness against external and internal attacks. While most implementations make the data itself immutable, that is not the same as secure. Ethereum has experienced numerous attacks over the past years, meaning it is difficult to consider it secure by default. On the contrary, Ethereum is more stable against internal attacks, excluding the possibility for majority of nodes to unite and and sophisticate the network, while such internal attack is possible in MultiChain.

3.2 Architecture

Our benchmarking tool (BPChain) consists of a web interface, a backend server, and a blockchain layer. The Web-interface communicates with the server, requesting the information from chosen blockchains and showing the benchmarking results to the user. The backend server consists of a database for storage and aggregation of blockchain data and controllers to ensure communication between frontend and blockchain layer. The Chain data collector receives information from all blockchain nodes of the different private chains on the FSOC server. Users can configure private networks, which is handled by Private Chain configurator component. Each provided blockcain implementation provides a master node distributing the scenario to all participating slave nodes (i.e. the nodes executing the scenario). This provides a layer of abstraction, so that additional blockchain implementations can be added with relative ease. They communicate with the FSOC Server through WebSockets. Both frontend and backend are running on a separate server inside Docker containers, for easier setup and deployment.

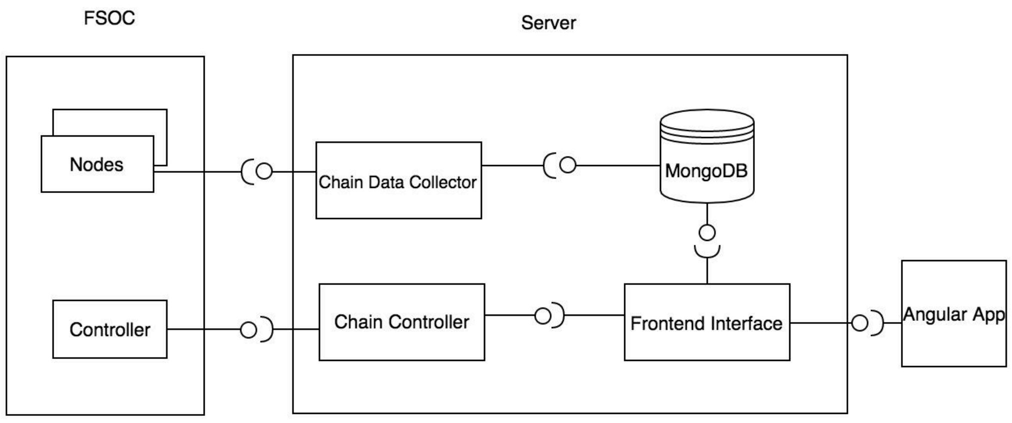


Figure 1: Solution architecture

3.3 Test Scenarios

When comparing blockchain protocols and the runtime properties they exhibit, the executed scenario (i.e. the enacted business process) heavily influences the measurable metrics. To allow benchmarking for a wide array of possible use cases, we have implemented two possibilities to define and enact scenarios on the different blockchains.

The first approach consist on defining a scenario based on the number of participating nodes, the interval with which each node sends a transaction, and the amount of data a single transaction contains. This allows to benchmark simple scenarios and provides a layer of abstraction regarding the intended use case of the blockchain platform. Each possible application running on a blockchain can be reduced to transactions with certain sizes being sent in specific intervals.

The second approach consists of modelling a use case using Business Process Model and Notation[4] (BPMN) and enriching the model with runtime information generated by Scylla [6], an extensible BPMN process simulator. This allows to benchmark complex scenarios in which the payload size for transactions and the interval between scenarios varies over time. Under the hood, this too is implemented by triggering transactions with certain sizes in certain intervals.

3.4 Used FSOC Resources

All experiments were executed on a Fujitsu RX600S5-2 architecture with 4 Xeon X7550 processors, each having eight cores with 2GHz, providing 1024 GB of RAM and running Java 7 and Ubuntu 14.04.
We built private Ethereum, XAIN, and MultiChain networks, which are deployed to the FSOC machines. Every node of each network runs in a separate Docker container, which allows to scale the network easily by incrementing the number of Docker containers. The provided resources make it possible to compare performance of networks running scenarios as described above.

3.5 Findings

The derived results were based on a test scenario and data elicited during experiments to support various bachelor theses. All researched blockchains were capable to execute a wide array of scenarios, but the experiment produced different results for each network — when executing different scenarios, the blockchain most suitable regarding a specific metric differed depending on the setup and the number of participating nodes, only further underlining the need for benchmarking and comparing different blockchains regarding their suitability for a concrete scenario.

4 Conclusions

The platform we built with the FSOC Lab resources delivers in-depth evaluation possibilities of three blockchain protocols. It is possible for user to set parameters and complex scenarios to analyse specific setups more precisely. We chose following parameters to compare networks on similar characteristics: a number of hosts and a number of miners in the network, an average block time, a difficulty of network, and a hash rate. Further metrics derived by these parameters, such as stability, data transfer, and energy consumption allow for a thorough analysis of suitability regarding a concrete scenario and several blockchains. Besides, we tested the scalability of all networks continuously increasing the number of nodes in each network. We concluded, that PoW networks are harder to scale as their speed depends on the number of nodes and transactions. At the same time networks using more resources friendly consensus protocols show better performance. The capabilities of the FSOC Lab were overall sufficient to conduct the tests. The blockchain benchmarking platform BPChain satisfied all requirements and is capable of providing suitable information to choose a blockchain implementation for a specific and concrete use case.

References

[1] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan. *BLOCKBENCH*. May 2017. DOI: 10.1145/3035918.3064033. arXiv: 1703.04057 [cs.DB].

[2] Greenspan, Gideon. *MultiChain Private Blockchain — White Paper*. URL: https://www.multichain.com/download/MultiChain-White-Paper.pdf (last accessed 2018-04-01).

[3] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf (last accessed 2018-04-01).

[4] Object Management Group. *Business Process Model and Notation*. 2018. URL: http://www.bpmn.org (last accessed 2018-04-23).

[5] R. Patel. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2018. URL: https://github.com/ethereum/wiki/wiki/White-Paper (last accessed 2018-04-01).

[6] L. Pufahl and M. Weske. "Extensible BPMN process simulator". In: URL: https://bpt.hpi.uni-potsdam.de/Public/scylla.

[7] A. Rosic. *Blockchain Scalability: When, where, how?* URL: https://blockgeeks.com/guides/blockchain-scalability/ (last accessed 2018-04-01).

[8] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A. B. Tran, and P. Rimba. *On Availability for Blockchain-Based Systems*. Sept. 2017. DOI: 10.1109/srds.2017.15.

[9] XAIN. *Practical Proof of Kernel Work & Distributed Adaptiveness*. URL: https://www.xain.io/pdf/XAIN-Yellow-Paper.pdf (last accessed 2018-04-01).

[10] R. Yasaweerasinghelage, M. Staples, and I. Weber. *Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation*. Apr. 2017. DOI: 10.1109/icsa.2017.22.

High-Performance Ensembles of Online Sequential Extreme Learning Machine for Regression and Time Series Forecasting

Luís Fernando L. Grim[1] and André Leon S. Gradvohl[2]

[1] Federal Institute of Education Science and Technology of São Paulo
lgrim@ifsp.edu.br
[2] High Performance Intelligent Decision Systems
School of Technology, University of Campinas
gradvohl@ft.unicamp.br

Ensembles of Online Sequential Extreme Learning Machine algorithm are suitable for forecasting Data Streams with Concept Drifts. Nevertheless, data streams forecasting require high-performance implementations due to the high incoming samples rate. In this work, we proposed to tune-up three ensembles, which operates with the Online Sequential Extreme Learning Machine, using high-performance techniques. We reimplemented them in the C programming language with Intel MKL and MPI libraries. In summary, our proposal consists of a two-level parallelization, where we allocated each ensemble model into an MPI process, and we parallelized the internal functions of each model in a set of threads through Intel MKL. Experimental results showed that, in general, the high-performance ensembles improve the execution time, when compared with its serial version, performing up to 10-fold faster.

1 Introduction

Several sectors related to the environment have used the data mining and machine learning tools, as in the predictions related to air quality [1]. We can consider some of these datasets as Time Series when they correspond to a collection of values obtained periodically. Also, we view these datasets as Data Streams, since they are an ordered sequence of instances arriving at high throughput, i.e. a high number of instances per unit of time. Considering these characteristics, the systems that deal with data streams cannot handle many instances in main memory for an extended period, or they may experience a memory overload.

Data streams are dynamic by nature, i.e. the underlying data distributions of these streams tend to change over time. This phenomenon, known in the literature as Concept Drift, consists in changing the relationship between the input data and the target variable that someone would like to predict over time [3]. This dynamic nature may compromise the model performance when predicting new incoming samples. Online learning approaches can deal with concept drift scenarios, due to the capability to learn new data patterns over time. Therefore, online learning is suitable for data streams' predictions.

Liang et al. [5] proposed the Online Sequential Extreme Learning Machine (OS-ELM) that updates the forecasting model according to the new instances' arrival. These characteristics make the OS-ELM able to handle concept drift implicitly, turning it suitable for the data streams processing. Besides, several ensembles in the literature can operate with OS-ELM models [1, 4, 6]. The ensembles of OS-ELM models tend to present improvements in stability and accuracy of the predictions. However, the use of ensembles also tends to increase execution time.

In this paper, we proposed to tune-up three ensembles, which operates with OS-ELM, using high-performance computing techniques, running in parallel in multicore CPUs and also in clusters with several computer nodes. Therefore, our goal is to verify if our proposals provide a significant improvement in execution time when compared to the respective conventional serial approaches.

2 Implemented Approaches

To adapt our proposed implementations to data stream processing, we considered a dataset with an N_0 size chunk for the initial training, and the remaining instances simulate an online stream. We also add a mechanism which enables OS-ELM operates using a Sliding Window (SW) of fixed size m which is used to update the model on the online step. In this work, we restricted the OSELMsw to the cases of regression with the sigmoid activation function ($g(x) = \frac{1}{(1+e^{(-x)})}$).

Using the Intel MKL, we can set a number of threads N_{th} to parallelize the OS-ELMsw functions. The Intel MKL enable us to perform the matrix operations using Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK) functions that explore the multithread features in multicore processors.

2.1 Ensembles of OS-ELMsw

We consider the "Ensemble of Online Sequential Extreme Learning Machine" (EOS-ELM) [4], the "Dynamic and On-line Ensemble Regression" (DOER) [6] and the "Ensemble of Online Learners with Substitution of Models" (EOS) [1] and reimplemented them in two versions, one with MPI standard and another serial. MPI allows us to parallelize tasks with distributed memory on several processes, which can be allocated within a single computational node, or spread over several nodes. In summary, our proposal consists of a two-level parallelization, allocating each ensemble model into an MPI process, parallelizing the internal functions of each model in a set of threads.

With this concept, in the high-performance versions with MPI (hpEOS-ELM, hpEOS, and hpSCDOER), each model can perform its prediction without waiting for the others. In other words, we run in parallel all the models through the ensemble at the same time. However, since they are online predictive ensembles, they need to be synchronized before receiving data from each block of instances. There is also a master-process to do some tasks that require inter-models communications.

3 Experiments Design

We adopted the synthetic Hyperplane dataset (HYP) using the Hyperplane generator, provided by Soares and Araújo [7], to get a stream with 175,000 instances. We also used a real dataset containing time samples of PM_{10} concentration collected sequentially from January 1, 1998, to November 23, 2017, from the Cubatão-Vila Parisi automatic station provided by the QUALAR system [2].

The PM_{10} dataset comprises 174,397 samples, displays zero as the minimum value, 1,470 $\mu m/m^3$ as the maximum value and 8,011 missing samples. To mitigate the effects of these missing samples, we fill in the most likely value using "Amelia II" and linear interpolation. In the PM_{10} dataset, we also performed an outliers analysis and normalized the data between $[0, 1]$. After that, we organized time samples such that we used the current sample and the last five instants of time, $x_n = [s_{n-5} \cdots, s_{n-1}, s_n]$, to predict the next instant $y = [s_{n+1}]$.

We tested cases with 100 hidden neurons and with update blocks of 1 instance. For the initial training, we used the 5,000 oldest instances, and we simulated a data flow with the remaining instances. We set the α of SCDOERs as 0.04. Based on previous results, we adopted a Sliding Window (SW) with 50 instances and the execution with 4 threads for Hyperplane dataset and a SW with 1 instance and the execution with 1 thread for PM_{10} dataset.

We run all tests in the 1000 Core Cluster from HPI Future SOC Lab, using the Slurm 17.02.6 and the Libraries Intel MKL 2018.2.199, and MPICH 3.2 installed.

4 Experimental Results

We repeated the experiments 20 trials and tested the performance considering the means and the standard deviations (SD). We evaluated the Total Real Time (TRT) of execution.

Tab. 1 gives the TRT of execution to all ensemble versions and Fig. 1 shows the high-performance versions speedups related to their serials. In versions with the suffix "nd", we allocated each MPI process on a distinct computing node. The other versions run within a single computing node.

Tab. 1 and Fig. 1 show that all high-performance versions ("hp" versions) presented improvements on TRT of execution in the cases related to Hyperplane dataset, performing up to 10-fold faster for the hpEOS-ELM with 8 models. For the PM_{10} dataset, all high-performance versions that we spawned into different computing nodes ("nd" versions) were slower than their serial versions, except for the hpEOSnd with 8 models. These results occur due to the delay in communication over the 10 Gigabit network between computing nodes. In all cases, the "nd" versions were also slower than the respective "hp" versions.

In both datasets, the hpEOS-ELM presented the highest accelerations. This behavior is related to the dynamics of operation, to the tasks, and to the transfers related to the master process of each ensemble. The hpEOS-ELM is a static ensemble, it runs with the same all models in parallel from the beginning, and it is also the one

Table 1: TRT of execution (in seconds) for the ensembles

Dataset	Ensemble	# of models		
		2	4	8
Hyperplane	EOS-ELM	266.70 ± 2.92	535.73 ± 4.42	1069.36 ± 7.49
	hpEOS-ELM	126.49 ± 3.32	94.56 ± 4.80	104.94 ± 8.16
	hpEOS-ELMnd	165.73 ± 5.17	179.74 ± 5.07	200.00 ± 5.77
	EOS	291.25 ± 1.96	569.58 ± 2.71	1091.72 ± 5.83
	hpEOS	142.80 ± 5.29	119.93 ± 5.75	141.73 ± 3.76
	hpEOSnd	182.07 ± 4.66	205.70 ± 5.76	234.76 ± 3.95
	SCDOER	328.12 ± 2.14	639.30 ± 4.80	1254.89 ± 6.75
	hpSCDOER	177.66 ± 4.57	185.95 ± 4.46	203.58 ± 5.94
	hpSCDOERnd	230.66 ± 2.69	283.21 ± 5.73	340.72 ± 7.58
PM_{10}	EOS-ELM	10.44 ± 0.06	20.03 ± 0.21	38.44 ± 0.36
	hpEOS-ELM	6.07 ± 0.09	6.58 ± 0.07	7.07 ± 0.10
	hpEOS-ELMnd	23.92 ± 1.94	43.52 ± 5.10	65.44 ± 3.56
	EOS	23.79 ± 0.15	65.44 ± 0.71	166.30 ± 1.35
	hpEOS	18.56 ± 0.36	30.44 ± 0.15	39.41 ± 0.15
	hpEOSnd	42.67 ± 3.42	66.92 ± 2.31	96.25 ± 2.65
	SCDOER	848.90 ± 2.47	875.35 ± 4.77	883.20 ± 3.02
	hpSCDOER	844.10 ± 2.28	837.65 ± 1.18	847.81 ± 1.14
	hpSCDOERnd	893.33 ± 3.52	935.56 ± 7.07	989.76 ± 7.16

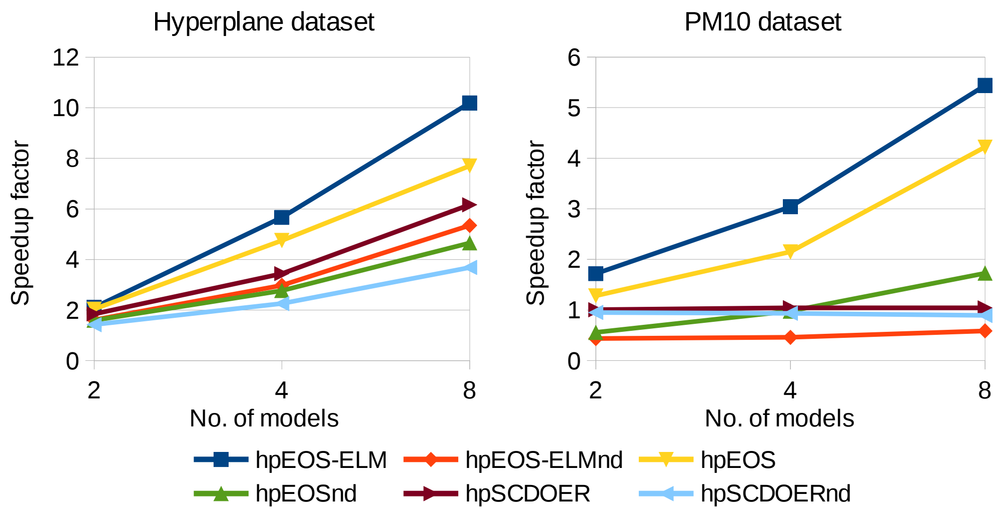


Figure 1: Speedups of high-performance versions related to their serial versions

that represents fewer tasks and data transfers to the master process. The hpEOS and hpSCDOER are incremental and dynamic ensembles. They start running with one model and adding others according to the rules of inclusion.

5 Conclusions

The results showed that our high-performance ensembles perform better when compared with their corresponding serial version in most cases. In general, the hpEOS-ELM presented the highest accelerations because of their simplicity and to the fact that they run all models in parallel from the beginning, unlike hpEOS and hpSCDOER, which are incremental.

Regarding the multi-node approaches, it would be interesting to perform tests in environments with high-performance networks like RDMA/InfiniBand. For single-node approaches, someone could test them on common multicore CPUs.

Based on the results obtained, we believe that the high-performance techniques presented in this work may also be suitable for other types of ensembles and online learning models. An interesting challenge would be the application in ensembles composed of models with adaptive sliding window.

References

[1] A. Bueno, G. P. Coelho, and J. R. Bertini. “Online Sequential Learning based on Extreme Learning Machines for Particulate Matter Forecasting”. In: *Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2017, pages 169–174. ISBN: 978-1-5386-2407-4. DOI: 10.1109/BRACIS.2017.25.

[2] CETESB. *Poluentes | Qualidade do Ar - Sistema Ambiental Paulista - Governo de SP*. 2017. URL: http://ar.cetesb.sp.gov.br/poluentes/ (last accessed 2017-03-29).

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. “A survey on concept drift adaptation”. In: *Computing Surveys* 46.4 (2014), pages 1–37. ISSN: 03600300. DOI: 10.1145/2523813. arXiv: 1010.4784.

[4] Y. Lan, Y. C. Soh, and G. B. Huang. “Ensemble of online sequential extreme learning machine”. In: *Neurocomputing* 72.13-15 (2009), pages 3391–3395. ISSN: 09252312. DOI: 10.1016/j.neucom.2009.02.013.

[5] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. “A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks”. In: *IEEE Transactions on Neural Networks* 17.6 (2006), pages 1411–1423. ISSN: 1045-9227. DOI: 10.1109/TNN.2006.880583.

[6] S. G. Soares and R. Araújo. “A dynamic and on-line ensemble regression for changing environments”. In: *Expert Systems with Applications* 42.6 (2015), pages 2935–2948. ISSN: 09574174. DOI: 10.1016/j.eswa.2014.11.053.

[7] S. G. Soares and R. Araújo. “An On-line Weighted Ensemble of Regressor Models to Handle Concept Drifts”. In: *Engineering Applications of Artificial Intelligence* 37 (2015), pages 392–406. DOI: 10.1016/j.engappai.2014.10.003.

Pychainlab: Analysing the Ethereum Network

A Framework to Model and Simulate a Blockchain Environment

Alexander Mühle, Andreas Grüner, Vageesh Saxena, and Tatiana Gayvoronskaya

Internet Technologies and Systems
Hasso Plattner Institute for Digital Engineering
{alexander.muehle,andreas.gruener,tatiana.gayvoronskaya}@hpi.de
saxena@uni-potsdam.de

Blockchain networks such as Bitcoin and Ethereum rely on Peer-to-Peer networking to propagate essential information such as transactions and blocks throughout the network. Testing the effect of latency and other network properties currently requires either analysis of a running real world network or simplified modelling for simulation. In this project we developed tools to run a Blockchain network (Ethereum) in a virtual environment (Docker) and adapt network properties such as latency, loss, bandwidth and jitter as needed. The resulting network is observed by logging important events and calculating parameters of interest such as propagation times for transactions and blocks.

1 Introduction

In 2008 Satoshi Nakamoto published the original paper on Bitcoin [9] and layed the foundation for the rise of cryptocurrencies and their underlying Blockchain technology.

A Blockchain is a distributed database comprised of a chain of blocks and maintained by nodes participating in a peer-to-peer network [6]. Each node of the peer-to-peer network holds a complete replica of the database. Transactions issued to the Blockchain network update the database based on a consensus algorithm. The consensus algorithm provides rules that are applied by the nodes to agree on the validity of Blockchain updates.

Following the success and popularity of Bitcoin, numerous alternative Blockchains were created using Bitcoin as foundation. However, these alternative Blockchains deviate from Bitcoin by modifying specific parameters, e. g. on peer-to-peer network, block or consensus level, to achieve different characteristics. Besides that, distinct general or specific purpose Blockchains are developed comprising an increased feature set compared to Bitcoin. The alternative Blockchain that we will focus on in this paper is Ethereum [15]. While Bitcoin is mostly talked about in the context of digital cash, Ethereum aims to provide a platform for a number of applications. The concept of Smart Contracts [13] allows for verifiable code execution of a (nearly) Turing-complete programming language in the Ethereum network.

Generally a Blockchain and in this case specifically Ethereum is a globally distributed network of nodes. Additionally, with Ethereum being a public and unpermissioned Blockchain, nodes can freely join and leave the network.

In this project we developed tools to help analyse the effects that changes in the basic parameters blocksize and blocktime/difficulty have on the Ethereum protocol and its consensus process, as well as the impact network properties such as latency and bandwidth have, while allowing arbitrary network constellations.

The remainder of this paper is organised as follows. We will first give an overview of related work on the analysis of Blockchain systems. Subsequently, our testing environment will be described in Section 3. In Section 4, we will discuss the parameters that are of interest in this study as well as how we defined them. The testing itself and its results are presented in Section 5. Finally, we will discuss future work and conclude our paper in Section 6 and 7 respectively.

2 Related Work

The main focus of related work so far has been on the Bitcoin network. In general, there have been two different approaches to analysing Bitcoin network behaviour. The most common approach has been to utilise network sniffers and analyse the real Blockchain networks. Donet et al. [3] utilised a modified Bitcoin P2P network sniffer [1] to identifiy the size as well as the geopgrahic distribution of the Bitcoin network. In the paper "Discovering bitcoin's public topology and influential nodes" Miller et al. [8] went into even more detail when analysing the distribution, size and influential operators of the network. This was also achieved using a network sniffer called CoinScope [7].

Donet et al. additionally analysed the availability of the discovered nodes as well as the propagation time of broadcasts in the network. Similarely Decker and Wattenhofer [2] focused on the properties of the peer-to-peer broadcast that is used in Bitcoin. They gathered information using strategically placed sniffers just as Donet et al. With this they gathered information on the propagation time of both transactions and blocks and identified this propagation delay as primary reason for blockchain forks, the inconsistencies in the replicated database of a Blockchain network. One problem in their measurement was the non-trivial clock skew in the timestamps. They found the actual clock skew to be quite pronounced, so much so that it produced impossible block constellations (out of order). They tried to mitigate this by observing the first occurence of a block in the network and use that time as substitute for the block's own timestamp. It has to be noted however that these measurements, while not suffering from clock skew, can be inaccurate as well due to the limited view of the network by the measurement nodes/sniffers. The same methodology has been used by Pappalardo et al. [10] when measuring the propagation dynamics of transaction and blocks as well as the transaction inclusion time.

The projects mentioned above focus on existing networks and their real life topology and performance. More in line with the intentions of this paper was the work by Gervais et al. "On the security and performance of proof of work blockchains" [5].

Here the impact of different consensus parameters and network properties on the security and performance was analysed. For this purpose Gervais et al. introduced a "Bitcoin Simulator" [4]. The Bitcoin Simulator was built on top of the ns3 network simulator and modelled the behaviour of the Bitcoin blockchain in regards to Proof of Work mining and the propagation of blocks. This left out the aspect of transaction propagation. In order to simulate the network itself (for block propagation) realisticly Gervais et al. relied on latency statistics provided by Verizon [14], bandwidth distribution by testmy.net and Bitcoin node distribution by bitnodes.earn.com.

3 Simulation Framework

Instead of participating in the public Ethereum network we opted to run a realistic network with the help of the FutureSOC Lab. This enables us to change network properties as needed as well as controlling all nodes in the network. However to stay flexible and allow real world implementations of Blockchain systems to be tested, we opted against simulating it as Gervais et al. did. For this purpose we created docker images for the Python Ethereum implementation pyethapp and configured them to interact with each other over virtual network interfaces provided by docker. We then modified a version of Blockade, originally desgined as a utility for testing network failures and partitions in distributed applications [16] to configure the network properties, latency, bandwidth and loss through the use of Linux traffic control and netem. For this purpose we grouped nodes into regions and assigned them their own subnet. We extended Blockade with a filter functionality allowing us to apply latency and loss properties to connections between specific regions and inside a region. The Blockade software also already allowed us to create network partitions as well as selectively let specific docker nodes join or leave the network. In order to capture accurate results we also extended the Python Ethereum implementation with remote logging capabilities, allowing us to gather all relevant logs at one Syslog server.

4 Methodology

In this section we will look at how the propagation information is gathered from the network. This includes adding remote logging capabilities to an Ethereum client. Finally we will present our preliminary experiment setup.

4.1 Logging & Parameter Calculation

We chose the Ethereum Python implementation *pyethapp* [11] for this purpose. Following log events have been added: T1 - transaction received, T2 - transaction distributed, B1 - block received, B2 - block distributed, B3 - block created; C1, header changed.

Transaction events include the IP and port of the peer that the transaction is sent to or received from. Additionally, it also includes the hash of the transaction as well as the sender, receiver and value of the transaction itself.

Block events include the IP and port of the peer that the block originated from or is sent to but more imortantly the hash of the block and hash of the previous block as well as all included transactions as hash.

The header change event signals that a specific node has replaced their head with a new block. The block information, hash, previous hash and transaction hashes of the replacement block are included in the event message.

4.1.1 Transaction and Block Propagation

The propagation of transactions and blocks is measured from the moment they are created by the originator node, indicated by the B3 and T2 event, until they are propagated to all nodes in the network. This means all nodes in the network have signalled a B2 / T1 event.

4.2 Experiment Setup

In our preliminary experiment we limited the network to only 15 nodes distributed according to ethernodes.org over 4 main regions, namely Europe, North America, South America and Asia. The network latency has been derived from Verizon statistics in line with Gervais et al. Figure 1 shows the resulting topology in more detail.

5 Results

We were able to produce some first results in the form of limited experiment runs with the previously described experiment setup (see table 1). It is however not a representative experiment yet as the number of nodes as well as number of connections each node makes is an important factor for transaction/block propagation.

Table 1: Transaction propagation and Block propagation relative to latency

Latency Multiplier	Transaction Propagation (ms)	Block Propagation (ms)
0	59	5508
0.5	521	15 079
1	1540	18 261

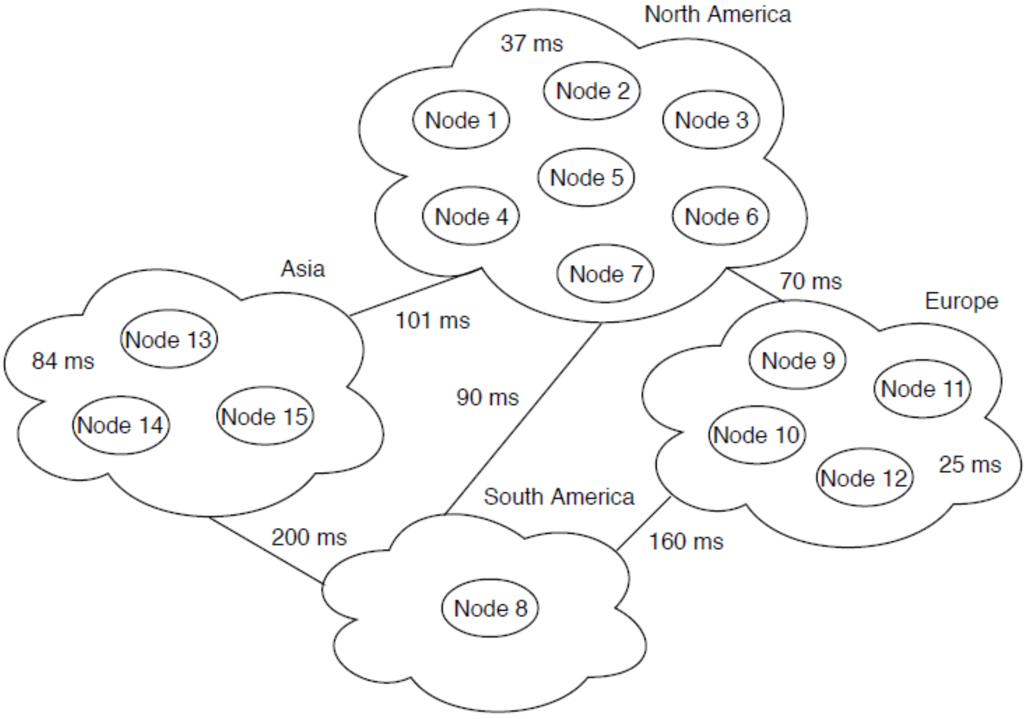


Figure 1: Experiment Setup Network Latencies

6 Future Work

In a next step we want to repeat this experiment with a more representative node count and node degree. However analysis of other parameters are of interest as well. As Blockchain networks are comprised of peer-to-peer communication it is of interest to us how efficient those networks are, considering different approaches to the construction of the peer-to-peer overlay networks used in Blockchain applications. In a next step we want to analyse the amount of inefficiency introduced by the topology mismatch problem in current Blockchain applications, as it has been recognised as efficiency problem in other popular peer-to-peer applications such as Gnutella [12]

7 Conclusion

In this project we developed tools to create a virtual Blockchain network and the ability to control network properties as needed. The resulting Blockchain network on the basis of the Python Ethereum implementation pyethapp can be logged and analysed by calculating important parameters such as transaction and block propagation. First preliminary results demonstrate the functionality of the tools.

References

[1] S. Castro. *Bitcoin P2P Network Sniffer v. 0.0. 2.* 2017. URL: https://github.com/sebicas/bitcoin-sniffer (last accessed 2018-04-01).

[2] C. Decker and R. Wattenhofer. "Information propagation in the bitcoin network". In: *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE. 2013, pages 1–10.

[3] J. A. D. Donet, C. Perez-Sola, and J. Herrera-Joancomarti. "The bitcoin P2P network". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pages 87–102.

[4] A. Gervais. *Bitcoin Simulator*. 2016. URL: https://github.com/arthurgervais/Bitcoin-Simulator (last accessed 2018-04-01).

[5] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. "On the security and performance of proof of work blockchains". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pages 3–16.

[6] C. Meinel, T. Gayvoronskaya, and M. Schnjakin. *Blockchain: Hype oder innovation*. Hasso-Plattner-Institute, 2018.

[7] A. Miller. *CoinScope*. 2015. URL: https://www.cs.umd.edu/projects/coinscope/ (last accessed 2018-04-01).

[8] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. "Discovering bitcoin's public topology and influential nodes". In: *et al.* (2015).

[9] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: (2008).

[10] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste. "Blockchain inefficiency in the Bitcoin peers network". In: *EPJ Data Science* 7.1 (2018), page 30.

[11] *pyethapp*. URL: https://github.com/ethereum/pyethapp (last accessed 2018-04-01).

[12] M. Ripeanu, A. Iamnitchi, and I. Foster. "Mapping the gnutella network". In: *IEEE Internet Computing* 1 (2002), pages 50–57.

[13] N. Szabo. "The idea of smart contracts". In: *Nick Szabo's Papers and Concise Tutorials* 6 (1997).

[14] Verizon. *Verizon Latency*. 2018. URL: http://www.verizonenterprise.com/about/network/latency/ (last accessed 2018-04-01).

[15] G. Wood. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151 (2014), pages 1–32.

[16] Worstcase. *Blockade*. URL: https://github.com/worstcase/blockade (last accessed 2018-04-01).

A Big Data Science Experiment
Identity deception on social media platforms

Estée van der Walt and Jan H.P. Eloff

Department of Computer Science
Security & Data Science Research Group
University of Pretoria, South Africa
estee.vanderwalt@gmail.com,eloff@cs.up.ac.za

In the past, systems and organizations at large were vulnerable to online attacks from various threat actors. More recently, these attacks have become targeted at individuals with Social Media Platforms (SMPs) being the enabler. Examples of such threats are cyber bullying, grooming, and phishing. These threats originate from non-human, also known as bots, or human online accounts. Related research has invested much effort in detecting attacks from bots on SMPs. This research proposes to focus on finding deceptive individuals instead. Past work from the researchers identified various features using bot related research as well as psychology with which humans, who potentially lie about their online identity, could be identified. These features, together with supervised machine learning algorithms, were evaluated using a dataset of online accounts from Twitter. The results showed that identity deception by humans could be detected but lacked interpretability. The current work continues by presenting an Identity Deception Detection Model (IDDM). The IDDM not only uses previous machine learning results, but also presents an interpretation of the results per individual. This information explains why a particular individual was perceived as being deceptive. This information could also further be used by SMPs to implement measures towards protecting vulnerable individuals against attacks from other humans.

1 Project idea

Many examples of Identity Deception exist on SPMs. Take for example a woman in Great Britain who was jailed for grooming a boy with a fake identity on Facebook [1]. Not all fake identities are harmful [3] but there are those fake identities that are used for malicious purposes like cyber-bullying, grooming, and phishing [4]. The researchers have shown in previous work [12] that features can be engineered given related work in bot detection and psychology. These engineered features, together with supervised machine learning can be used towards the detection of human identity deception.

Although past research was successful, the results did not provide any further information as to why a specific individual would be perceived as being deceptive. This is a well-known problem in machine learning where machine learning models are still being regarded as so-called *black box* [5]. This means that the results from the machine learning model does not provide further information as to how it reached its final decision.

For this research, related work in machine learning model interpretation were first evaluated to understand the scope of the problem. It was found that decision tree algorithms are proposed to be more interpretable than other types of machine learning algorithms [2]. Saabas [9] reverse-engineered the random forest algorithm to produce a method, call 'tree interpreter'. Olah et al. [7] interpret the images recognised by a neural network machine learning model by showing how each layer of the neural network visually evolves. Ribeiro et al. [8] propose a method called 'LIME' (Local Interpretable Model-Agnostic Explanations). Lundberg and Lee [6] proposed that by using Shapely values to change feature weights, an explanation can be achieved that resembles human intuition. The computational overhead associated with the each of these interpretive calculations differs. LIME, for example, requires many iterations of models to be developed per SMP user to explain their deceptiveness.

This computational overhead needs to be considered in an environment like SMPs, where data is produced at high velocity. Another consideration is that related research applied to specific machine learning algorithms. Due to the computational overhead associated with related interpretation methods mentioned and the lack of a generic interpretative model, the researcher looked towards the use of Shannon's entropy [11] as a measure with which results can be explained. Entropy is calculated during the development time of supervised machine learning model and readily available to be applied to interpret the results.

1.1 Main deliverables

This research is a continuation of past work as illustrated in Figure 1. The main deliverables for this period were as follows:

- Present a model to assist in the automated detection of human identity deception.
- Build an interpretive identity deception score for each individual to explain the results of the model.

2 Use of HPI FutureSOC Lab resources

The following resources were used for the research at the HPI FutureSOC Lab for the purposes of developing and experimenting with the IDDM:

- Twitter: The Twitter4j Java API was used to mine the data needed for the experiment in a big data repository [13].
- SAP HANA [10]: An SAP HANA instance were used as data repository. The machine contains 4 TB of storage, 2 TB of RAM (1.4 TB effective) and 96 cores.

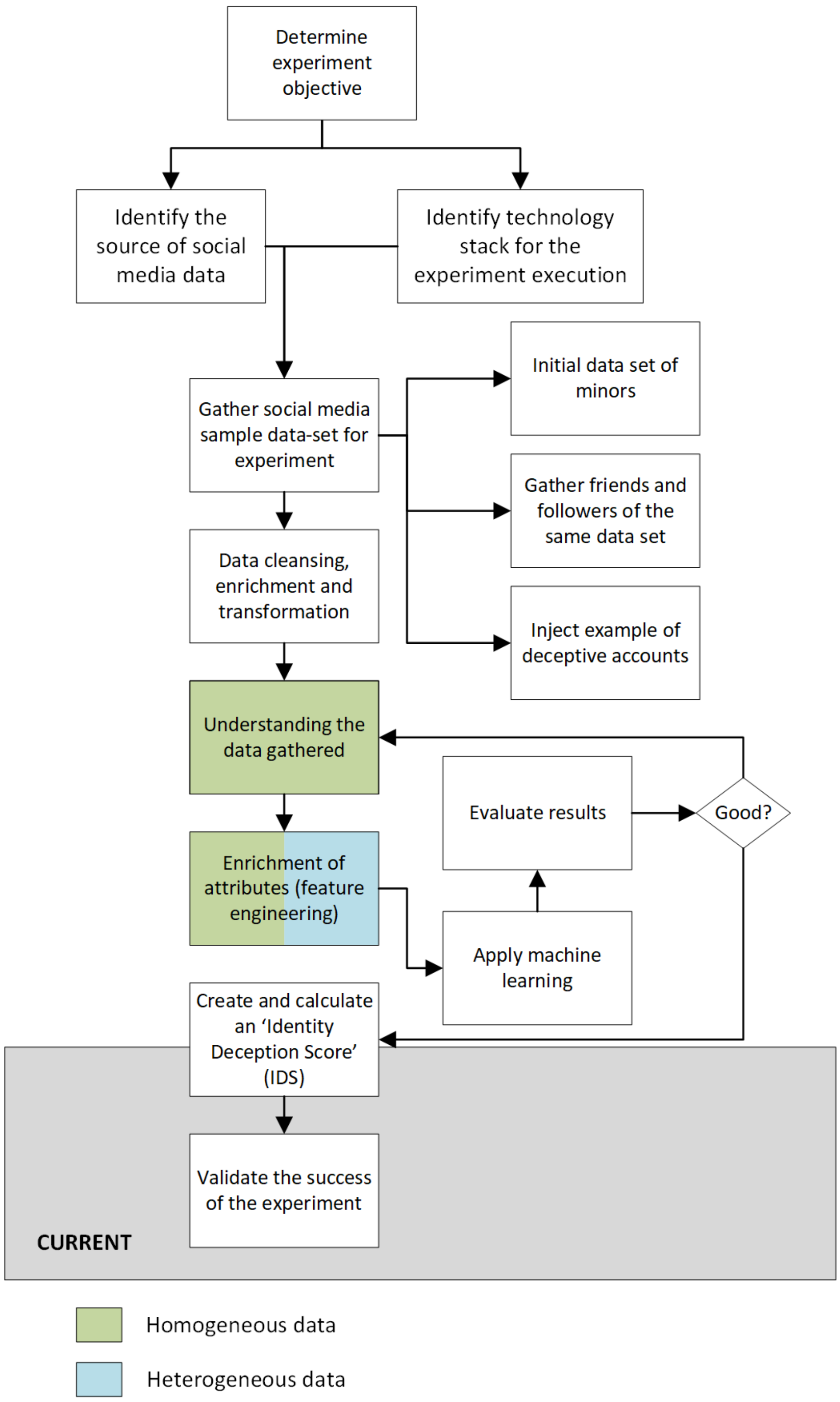

Determine experiment objective
Identify the source of social media data
Identify technology stack for the experiment execution
Gather social media sample data-set for experiment
Initial data set of minors
Gather friends and followers of the same data set
Inject example of deceptive accounts
Data cleansing, enrichment and transformation
Understanding the data gathered
Evaluate results
Good?
Enrichment of attributes (feature engineering)
Apply machine learning
Create and calculate an 'Identity Deception Score' (IDS)
Validate the success of the experiment
CURRENT
Homogeneous data
Heterogeneous data

Figure 1: Research steps

The in-memory high-performance processing capabilities of SAP HANA enables almost instantaneous results for analytics.

- Machine learning APIs: Various tools were considered to perform classification, analysis, and apply deep learning techniques on the data. These include the PAL library from SAP HANA, SciPy libraries in Python, and R. For the research, R was the final choice due to support on this platform and libraries freely being available on the web community at a large scale.
- An additional Linux machine was provided for the lab to aid in the running of the CPU and memory intensive machine learning algorithms. The VM has 8 cores and 64 GB of RAM.

Overall, we found that the environment and its power enabled the collection and handling of a big dataset without issue. The continued support of the HPI FutureSOC Lab was and is greatly appreciated.

3 Findings in the Spring 2018 semester

The Identity Deception Detection Model (IDDM) was proposed to assist in the automated detection and interpretation of human identity deception. The IDDM consists of two components:

- The Identity Deception Detection Machine Learning Model (IDDMLM) makes use of past experimental supervised machine learning results to predict whether new humans are perceived as deceptive.
- The Identity Deception Detection Scoring Model (IDDSM) takes the deceptive prediction, together with the entropy information that was calculated during earlier experimentation, to produce an interpretable deception score. Table 1 shows the IDDSM results for one such user (U_1).

Table 1: Results obtained from the IDDSM model

SMP Attributes and Features (A_3)	Explained (U_1)	Entropy
COMPARE_AGE	🟥	71.88
COMPARE_GENDER	🟥	21.09
DISTANCE_TIMEZONE	🟥	10.69
FOLLOWERS_COUNT	🟥	6.50
FRIENDS_COUNT	🟥	6.84
STATUS_COUNT	🟥	6.62
USERNAME_LENGTH	🟥	3.99

The results show that there were engineered features indicative of U_1 being perceived as deceptive. The individual's age and gender were the most relevant indicators. A further set of random IDDSM results were manually evaluated by the researchers. The results showed promise in explaining the potential deceptiveness of a human on an SMP like Twitter.

4 Future research and Conclusion

The requirement of the IDDSM component is to interpret the results from the IDDMLM. This is important for a number of reasons. *Firstly*, the interpretation explains how the model determines whether a human is deceptive or trustworthy. *Secondly*, the interpretation allows an agency, like law enforcement, to act on the results, and it highlights those humans who are most deceptive, with reasons as to why this decision was reached. *Thirdly*, this information can be used by SMPs themselves to improve their platform. If an SMP, for example, knows that location is an indicator of deception, they could implement additional measures to verify a person's location.

Future research proposes to add more engineered features indicative of human deceptiveness and to test the IDDM results against various known deceptive use cases found within SMPs. This will not only improve human identity deception detection success but also provide more feedback as to why particular individuals are perceived as being deceptive.

References

[1] B. P. Association. *Woman jailed for abusing boy after grooming him over Facebook*. URL: http://www.dailymail.co.uk/wires/pa/article-5206541/Woman-jailed-abusing-boy-grooming-Facebook.html (last accessed 2017-12-22).

[2] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller. "How to Explain Individual Classification Decisions". In: *Journal of Machine Learning Research* 11.61 (2010), pages 1803–1831. URL: http://jmlr.org/papers/v11/baehrens10a.html.

[3] L. Dedkova. "Stranger Is Not Always Danger: The Myth and Reality of Meetings with Online Strangers". In: *Living in the digital age* (2015), page 78. URL: http://www.ivdmr.fss.muni.cz/doc/living-in-da.pdf#page=79.

[4] M. A. Al-garadi, K. D. Varathan, and S. D. Ravana. "Cybercrime detection in online communications: The experimental case of cyberbullying detection in the Twitter network". In: *Computers in Human Behavior* 63 (2016), pages 433–443. ISSN: 0747-5632.

[5] Z. C. Lipton. "The mythos of model interpretability". In: *ICML Workshop on Human Interpretability in Machine Learning*. Series The mythos of model interpretability. 2016. arXiv: 1606.03490.

[6] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Volume 30. NIPS'17. Long Beach, California, USA: Curran Associates, Inc., 2017, pages 4768–4777. ISBN: 978-1-5108-6096-4. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[7] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. "The Building Blocks of Interpretability". In: *Distill* 3.3 (2018). ISSN: 2476-0757. DOI: 10.23915/distill.00010. URL: https://distill.pub/2018/building-blocks.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pages 1135–1144. ISBN: 1-4503-4232-9.

[9] A. Saabas. *Package for interpreting scikit-learn's decision tree and random forest predictions.* URL: https://pypi.org/project/treeinterpreter/ (last accessed 2018-04-07).

[10] SAP. *SAP HANA*. URL: https://www.sap.com/uk/developer/topics/sap-hana.html (last accessed 2018-08-31).

[11] C. E. Shannon. "A mathematical theory of communication". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), pages 3–55. ISSN: 1559-1662.

[12] E. van der Walt, J. Eloff, and J. Grobler. "Cyber-security: Identity Deception Detection on Social Media Platforms". In: *Computers & Security* (2018). ISSN: 0167-4048.

[13] Y. Yamamoto. *Twitter4J*. URL: http://twitter4j.org (last accessed 2018-05-16).

Strategies for an Improved GPU-Accelerated Skeleton Discovery for Gaussian Distribution Models

Christopher Schmidt, Johannes Hügle, and Matthias Uflacker

Enterprise Platform and Integration Concepts
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

The examination of causal graphical models supports the solution of important problems in a wide range of domains. For example in medicine, gene regulatory networks are a practical embodiment of systems biology and can be applied in diagnostics, or drug design. There exist several constraint-based algorithms for the derivation of these causal graphical models. All of them share the skeleton discovery of the well-known PC algorithm. In our work, we investigate different strategies to exploit the performance capabilities of GPUs to speed up the algorithm. Thereby, we aim to mitigate the performance issue due to the high computational complexity which is exponential to the dimension of the dataset in the worst case. We propose and investigate two strategies that extend our previous CUDA-based implementation for the skeleton discovery. They address shortcomings in the utilization of the GPU for higher order independence tests. An evaluation reveals that fusing multiple CI tests improves the performance by a factor of 1.77, as the parallel processing capabilities of the GPU are used more efficiently. Additionally, we found out that a concurrent execution of higher order independence tests does not improve the overall performance.

1 Introduction

The detection of causal relationships in complex systems from observational data allows to gain new insights, particularly in high-dimensional settings such as in genetics. In the context of causal graphical models, causal relationships are encoded in directed edges between vertices which represent the relevant variables in the system [17]. The construction of gene regulatory networks inferred from gene expressions are an example from genetics that enables to solve diverse biomedical problems in the context of diagnostics, or drug design [19].

In order to learn these causal structures, constraint-based algorithms, such as the PC algorithm developed by Spirtes et al. [23], conduct CI tests to derive the underlying structure of relationships. Building on this skeleton, the PC algorithm orients the detected undirected edges to construct a causal graphical model. Several extensions of the PC algorithm exist [5, 20, 23], which all employ the skeleton discovery of the PC algorithm. Thus, the results presented in this report apply to these extensions as well.

Due to its computational complexity, which may be exponential to the number of variables in the worst case, the application of the PC algorithm in practise is limited and remains a challenge for high-dimensional datasets such as in genetics [9]. To mitigate the long execution time for high dimensional datasets, we investigate the use of GPUs for the causal structure learning in the context of Gaussian distribution models. Devices, such as the NVIDIA K80 GPU, provided by the Future SOC Lab, reach a peak performance of up to 8.74 TFLOPS, are equipped with 24 GB of on-chip high bandwidth memory and outperform CPUs [15]. Leveraging the peak performance requires a suitable data parallel task, which follows the SIMT [11] execution model of a GPU.

In previous work, we have shown how the hardware capabilities of GPUs accelerate the first levels of the skeleton discovery [21]. In this project report, we extend this concept and focus on strategies to improve the utilization of the GPU for higher order CI tests. In particular, we target the skeleton discovery of the PC algorithm [23] for an underlying Gaussian distribution model. We explore CUDA's stream capabilities and a technique of fusing multiple CI tests to fully utilize the parallel processing capabilities of the GPU. Thereby, we aim to overcome performance bottlenecks for the CI tests for higher levels, as identified in our previous report.

The remainder of this report is organized as follows. Section 2 provides a detailed introduction into constraint-based causal structure learning in the context of the Gaussian distribution model. In Section 3, we present the implemented strategies for an improved CUDA-based implementation of the skeleton discovery, focusing on higher order CI tests. We provide preliminary experimental results in Section 4 and close our work with a summary in Section 5.

2 Causal Structure Learning

This section provides an introduction to the concepts of constraint-based causal structure learning with a focus on the order-independent version of the PC algorithm.

2.1 Preliminaries

A causal graph is a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ which consists of a finite set of N vertices $\mathbf{V} = (V_1, \dots, V_N)$, and a set of directed edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ that do not allow for any cycles. A directed edge $V_i \to V_j$ of a DAG defines a direct causal relationship from the variable V_i to the variable V_j, for $V_i, V_j \in \mathbf{V}$ [17, 23]. As several DAGs may express the same CI information they form a Markov equivalent class which can be uniquely characterized by a CPDAG [2, 3].

Therefore, the challenge is to estimate the equivalence class of the DAG $\mathcal{G}$ based on the joint probability distribution $\mathcal{P}$ of the involved variables $V_1, \dots, V_N$. On the assumption that the true causal DAG $\mathcal{G}$ generates the distribution $\mathcal{P}$, there exists an edge $V_i - V_j$ in the skeleton of $\mathcal{G}$ if and only if V_i, V_j are dependent given all sets $\mathbf{S} \subseteq \mathbf{V} \setminus \{V_i, V_j\}$ [23]. Hence, analyzing the CI information of the observed variables $V_1, \dots, V_N$ enables the estimation of the undirected skeleton $\mathcal{C}$ of the corresponding

DAG $\mathcal{G}$. A repeated execution of deterministic edge orientation rules on the skeleton $\mathcal{C}$ leads to the equivalence class of the underlying DAG $\mathcal{G}$ [4, 7, 22].

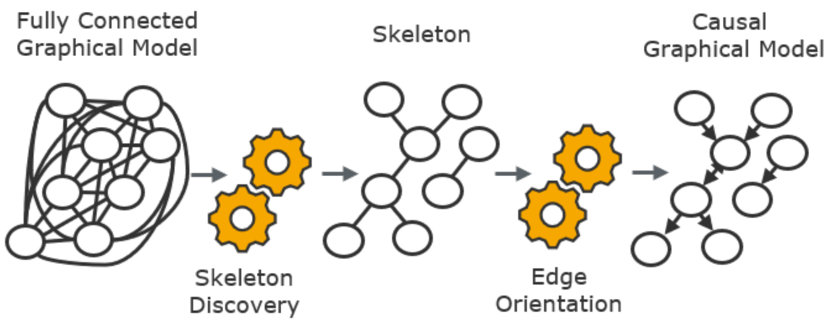


Figure 1: A schematic representation of the causal structure learning

This procedure for learning the causal graphical model was introduced by Spirtes et al. [23] and its schematic representation is depicted in Figure 1.

2.2 Gaussian Distribution Model

The joint distribution of the observed variables $V_1, \ldots, V_N$ determines the CI test to be applied during the skeleton discovery [6]. For gene expression data, the joint distribution distribution $\mathcal{P}$ is often a multivariate normal one which results in CI tests that are based on the partial correlations [12, 19].

In this Gaussian distribution model, two variables V_i and V_j are (conditionally) independent (given $\mathbf{S} \subseteq \mathbf{V} \setminus \{V_i, V_j\}$) if and only if the (partial) correlation coefficient ρ is equal to zero [1, 8, 24].

Therefore, the sample (partial) correlation coefficients provide the required (conditional) independence information that is needed in the skeleton discovery.

While the independence of two variables $V_i, V_j \in \mathbf{V}$ builds upon the well-known sample correlation coefficient $\hat{\rho}(V_i, V_j)$ of two variables $V_i, V_j \in \mathbf{V}$ the conditional independence is based in the sample partial correlation coefficient $\hat{\rho}(V_i, V_j | \mathbf{S})$. The sample correlation coefficient $\hat{\rho}(V_i, V_j | \mathbf{S})$ between V_i and V_j given a set of variables $\mathbf{S}$ can be obtained through the inversion of the corresponding sample correlation matrix $\mathrm{Cor}^{-1}(V_i, V_j, \mathbf{S})$, where $\mathrm{Cor}(V_i, V_j, \mathbf{S})_{k,l}$ is the sample correlation coefficient $\hat{\rho}$ for all $V_k, V_l \in \mathbf{S} \cup \{V_i, V_j\}$ [24]. On this basis, the sample partial correlation coefficient is determined by

$$\hat{\rho}(V_i, V_j | \mathbf{S}) = \frac{-r_{V_i,V_j}}{\sqrt{r_{V_i,V_i} r_{V_j,V_j}}}, \tag{1}$$

with r_{V_i,V_j} representing the corresponding element in the inverse sample correlation matrix $\mathrm{Cor}^{-1}(V_i, V_j, \mathbf{S})$. In case of singular or higher dimensional sample correlation matrices, the pseudo-inverse is computed using the Moore-Penrose generalized matrix inverse allowing for an accurate and robust calculation of the inverse [18].

Standard statistical hypothesis testing theory, e. g., see [10], is applied to test if the (partial) correlation is zero or not. Hence, using the corresponding sample (partial) correlation coefficient $\hat{\rho}$ the p-value is computed using

$$p = 2\left(1 - \Phi\left(\sqrt{n - |\mathbf{S}| - 3}\,|Z(\hat{\rho})|\right)\right), \tag{2}$$

where $\Phi(\cdot)$ stands for the cumulative distribution function of a standard normal distribution, and $Z(\cdot)$ stands for the Z-transformation of the corresponding estimated (partial) correlation coefficient $\hat{\rho}$. Hence, the null-hypothesis $\hat{\rho} = 0$ against the two sided alternative $\hat{\rho} \neq 0$ is rejected if for the corresponding p-value it holds that $p \leq \alpha$ where α denotes the selected significance level. Since many CI tests need to be performed, α should be interpreted as a tuning parameter, where smaller values of α tend to derive sparser graphs. In application the tuning parameter α is oftentimes set to equal 0.01, e. g., see [4].

2.3 Constraint-Based Skeleton Discovery

Using the concepts introduced in Subsection 2.1 we explain the PC algorithm by Spirtes et al. [23] in more detail. Particularly, we focus on the order-independent PC-stable version by Colombo et al. [4], which is sketched in Algorithm 1. Starting with a completely connected undirected skeleton $\mathcal{C}$ the PC-stable algorithm uses CI-tests given an increasing separation set $\mathbf{S}$ of adjacent vertices to reduce the number of edges in the skeleton $\mathcal{C}$.

Hence, only the CI tests of vertices V_i and V_j given separation sets $\mathbf{S}$ with size $l = 0$ up to the maximum size of adjacent vertices in the underlying DAG $\mathcal{G}$ need to be conducted (see lines 8-16 in Algorithm 1). This approach makes the algorithm computationally feasible, enabling its application in high-dimensional settings [7].

For every level l, the algorithm keeps the adjacency sets $a(V_i)$ of the variable V_i with regard to the current skeleton $\mathcal{C}$ (see lines 4-6). Based on this adjacency set, for every level $l = 0, \ldots, |a(V_i) \setminus \{V_j\}|$, all pairs of vertices V_i, V_j adjacent in $\mathcal{C}$ are tested for (conditional) independence with regard to a separation set $\mathbf{S}$ of size l by estimating the corresponding p-value and checking if $p \leq \alpha$ holds, or not. The edge $V_i - V_j$ is deleted from $\mathcal{C}$ and the according separation set $\mathbf{S}$ is saved in **Sepset**, in the case that the two variables V_i, V_j have been determined independent (see lines 11-13). Once, all pairs of adjacent vertices V_i and V_j of the current version of the skeleton $\mathcal{C}$ are dealt with, l is incremented by one. This process repeats until l reaches the maximum size of the adjacency sets of the vertices in the underlying DAG $\mathcal{G}$.

After completion the derived skeleton $\mathcal{C}$ and the information on conditional independence in **Sepset** form the input to compute the corresponding CPDAG through application of deterministic orientation rules, e. g., see [4, 7, 22].

3 Implementation of a GPU-Accelerated Skeleton Discovery

For the GPU-Accelerated implementation we use CUDA as the parallel computing platform and programming model [13]. Our CUDA-enabled implementation of the

Algorithm 1: Skeleton Discovery of PC-stable [4]

Input: Vertex set V, correlation matrix Cor
Output: Skeleton $\mathcal{C}$, separation sets **Sepset**

```
Start with fully connected skeleton $\mathcal{C}$ and $l = -1$
repeat
    $l = l + 1$
    for all Vertices $V_i$ in $\mathcal{C}$ do
        Set adjacent vertices of $V_i$ in $\mathcal{C}$ by $a(V_i)$;
    end for
    repeat
        Select $V_j \in a(V_i)$ with $|a(V_i) \setminus \{V_j\}| \geq l$
        repeat
            Choose $\mathbf{S} \subseteq a(V_i) \setminus \{V_j\}$ with $|\mathbf{S}| = l$.
            if for corresponding p $\leq \alpha$ then
                Delete edge $V_i - V_j$ from $\mathcal{C}$;
                Add S to Sepset;
            end if
        until edge $V_i - V_j$ is deleted in $\mathcal{C}$ or
        all $\mathbf{S} \subseteq a(V_i) \setminus \{V_j\}$ with $|\mathbf{S}| = l$ are chosen
    until all adjacent $V_i$, $V_j$ in $\mathcal{C}$ such that
    $|a(V_i) \setminus \{V_j\}| \geq l$ have been considered
until all $V_i$,$V_j$ adjacent in $\mathcal{C}$ satisfy $|a(V_i) \setminus \{V_j\}| \geq l$
return $\mathcal{C}$, Sepset
```

skeleton discovery algorithm is based on level-specific kernels, which enable optimizations for parallel processing CI tests on a GPU. Separate subsequently called kernels for the different levels l of the PC-stable algorithm, see Algorithm 1, also achieve the necessary synchronization between levels.

In previous work, we presented efficient CUDA-based implementations for levels $l = 0, 1$ [21]. Since the number of CI tests is known before the kernel for level $l = 0$ is launched, each CI test is processed by a separate thread. To conduct an independence test for V_i and V_j, the corresponding p-value is calculated via (2), see line 11 of Algorithm 1. The implementation makes use of existing functions from the CUDA Math API [14], including `sqrt()` and `normcdf()`.

In contrast the number of conducted CI tests in level $l = 1$ and level $l \geq 2$ is not known prior to the kernel launch. The number depends partially on the results of the previous level, as the adjacency sets $a(V_i)$ have changed and on the number of remaining candidates for a separation set.

Hence, the kernel for level $l = 1$, is launched with 32 threads, a single warp, per thread block. The threads within each thread block conduct all CI tests for any valid separation set S given the two vertices V_i, V_j. In order to determine the correct separation set S separating the vertices V_i, V_j the threads synchronize their results in shared memory. A single CI test is based on the inversion of the corresponding sample correlation matrix Cor, as explained in Subsection 2.2. As Cor is a 3×3-matrix for level $l = 1$, this can be done direct.

For level $l \geq 2$ the calculation of the pseudoinverse becomes more complex. As stated in our previous report, we compute the pseudoinverse via SVD followed by a matrix multiplication of U, $V^{**}T$ and S^{-1}, where U denotes the left-singular vectors, V the right-singular vectors and S the singular values of Cor. This is realized by using existing efficient implementations from the `cuSolverDN` and `cuBLAS` in addition to own CUDA kernel implementations, e. g., for inverting the singular values S. The chosen implementation strategy showed a significant shortcoming, as each CI test is conducted separately with individual kernel launches. This leads to poor utilization of the GPU. Since the size of the corresponding sample matrix Cor is determined by the current level l through $l + 2$, the individual kernel launches do not expose enough parallelism for the capabilities of a GPU. Furthermore, the implementation conducts the CI tests of a single level l with $l \geq 2$ in a sequential manner. In the following, we propose two strategies to address these inefficiencies.

3.1 Fused Higher Order Conditional Independence Test

In order to reach a better overall execution time for processing multiple CI tests for level $l \geq 2$, we propose a fused CI test. Given the small size of the sample matrix Cor for level $l \geq 2$, e. g., for $l = 2$ Cor is a 4×4 matrix, the measured warp efficiency, achieved occupancy, and SM-efficiency are low, e. g., compare the first entry in Table 1. Knowing that the CI test within a level l are independent from each other, thus can be calculated in parallel, we propose to fuse multiple CI tests into a single operation. Therefore, we change the alignment step, which produces the corresponding sample matrices Cor for each CI to create fused sample matrices Cor_{fused}. These contain multiple instances of Cor. Aligning the separate instances of Cor along the diagonal in Cor_{fused} allows for direct use of the same functions from the `cuSolverDN` and `cuBLAS` library. By avoiding any overlap of single instances of Cor within Cor_{fused} correct results are guaranteed. Note that fusing multiple CI tests increases the memory footprint quadratic, as the alignment along the diagonal requires padding with zeros to recreate a dense matrix.

3.2 Concurrent Higher Order Conditional Independence Test

Since fusing the CI tests increases the memory footprint, we consider another approach to improve the performance of our GPU-accelerated implementation for higher level $l \geq 2$. Given that the problem size for conducting CI tests stay relatively small, e.g., for $l = 2$ we process 4×4 matrices Cor, we know that the SMs on the GPU are underutilized. Therefore, we adapt our implementation to utilize the concept of CUDA Streams.[1] When launching kernels in separate CUDA Streams, they can be executed concurrently on the GPU.

To enable the Stream feature, our implementation is changed in the following way. First, we create a CPU thread for each edge that has to be tested for conditional independence. Given that the number of CI tests for each edge varies depending on the separation set candidates, we assume that we achieve a better balanced workload, compared to assigning several edges to a fixed number of threads upfront. By setting the compile flag `--default-stream` to `per-thread` we ensure that each CPU thread starts its own CUDA Stream. Next, each thread conducts the necessary CI tests for the edge to be processed and launches the according GPU kernels in its own CUDA Stream. Prior to calls to the `cuSolverDN` and `cuBLAS` library, we ensure the execution within a separate CUDA Stream, by setting it explicitly using `cusolverDnSetStream` and `cublasSetStream` respectively. The computed results are synchronized, prior to the start of the next level l.

4 Preliminary Results

We evaluate each improvement strategy for level $l \geq 2$ separately, focusing on limiting factors identified in the previous implementation, e.g., SM-efficiency, achieved occupancy and warp efficiency. Therefore, we profile several runs using `nvprof` [16], the profiling tool provided by NVIDIA. For the experiments shown in this report we limit ourselves to level $l = 2$. We experienced similar behaviour for higher level $l \geq 3$ with respect to the increased problem size. Furthermore, the number of CI tests conducted per level l in real world settings decreases in higher levels $l \geq 2$ [21]. For the measurements we use randomly generated data, satisfying the Gaussian distribution model, that guarantees the computation of valid results, e.g., during

[1]http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#streams-cdp (last accessed 2018-04-01).

pseudoinverse calculation. The experiments are executed on a system equipped with an NVIDIA K80 GPU, with CUDA version 9.2, as provided by the Future SOC Lab.

For our fused CI test implementation we used `nvprof` [16] to profile the kernel executions. Since conducting a single CI test invokes multiple different kernels, we report the SM-efficiency, the warp efficiency and the achieved occupancy weighted according to its activity on the GPU. The results of our profiling runs are presented

Table 1: Profiling the effect of fusing multiple CI test into a single operation

Number of CI tests fused into a single operation	% SM efficiency	% Warp efficiency	% Achieved occupancy
1	5.63	32.74	4.20
2	7.69	33.98	4.29
8	16.39	39.39	4.73
64	26.80	47.19	13.64
512	77.33	88.87	59.04
4096	94.76	96.59	84.06

in Table 1. Note, that fusing a single CI test represents an execution without fusing. With a larger number of fused CI tests we see that a higher utilization of the available GPU resources is achieved. We almost fully saturate the compute resources, when executing 4096 CI tests in a single fused operation.

While the profiling results seem promising for our proposed fusion approach, we have not yet investigated its impact on the execution time. For this purpose we conducted the following measurements. We assume a fixed number of 4096 CI tests that have to be processed within the current level $l = 2$. Starting without fusing we have a baseline and increase the number of CI tests fused into a single operation by powers of two, up to 4096, hence processing all CI tests in a single fused operation. Table 2 presents the results of this experiment, which shows that we can achieve a speed-up of a factor up to 1.77 when fusing 32 CI tests. Despite the high utilization of the GPU as shown in Table 1 for large numbers of fused CI tests, we experience a significant slow-down when fusing 512 and more CI tests. Based on additional profiling results, we account this slow-down to the significant increase of the memory footprint. For 32 fused CI tests, we create a matrix Cor_{fused} of size 128×128, requiring several kilobytes of memory, compared to a matrix Cor_{fused} of size 2048×2048 for 512 fused CI tests, which requires several megabytes of memory. Hence, the transfer time for data to the GPU is increased. Additionally, with smaller sizes of matrices the GPU caches are used more efficiently during the computation of the pseudoinverse and the matrix multiplications.

We conducted a similar experiment with a fixed number of 4096 CI tests to be processed for the concurrent execution strategy using CUDA Streams. We varied the number of threads, hence streams, utilized during processing, from 2 to 32. Profiling

Table 2: Average execution time for processing 4096 CI tests, when fusing multiple CI tests into a single operation

Number of CI tests fused into a single operation	Execution time (s)
1	11.46
2	9.63
4	7.68
8	7.2
16	6.9
32	6.49
64	6.61
128	6.71
256	9.85
512	22.58
1024	74.31
2048	291.11
4096	1149.49

the runs using `nvprof` [16] we ensured that the launched kernels are running concurrently. We noticed a slower execution time when using CUDA Streams compared to our baseline without CUDA Streams. Furthermore, we noticed a slight increase in execution time when using more CUDA Streams. We assume that the poor performance of our concurrent execution strategy using CUDA Streams is due to the larger number of very short kernels launched during execution. Nevertheless, further investigation is needed to confirm this assumption and overcome the shortcomings.

5 Summary

In this project report, we shared our work towards a GPU-Accelerated causal inference. After an overview on causal inference, the Gaussian distribution model and the underlying concepts, we presented two implemented strategies for higher order CI tests. These strategies, namely concurrent CI tests and fused CI tests, address shortcomings of our previous implementation. Preliminary evaluation results show that fusing CI tests can improve the execution time by a factor of up to 1.77. For the concurrent execution of CI tests we did not get any improvement at all, which requires further investigation. In future work, we plan to improve the fused CI test further. We encountered an increased memory footprint, which we aim to address. Therefore, we want to investigate if batched operations as provided by the `cuSolverDN` and `cuBLAS` libraries are useful or if we can utilize sparse matrix representations to reduce the memory footprint.

References

[1] H. Ahrens. "Dempster, A. P.: Elements of Continuous Multivariate Analysis. Addison-Wesley Publ. Co., Reading, Mass. 1969. XII, 388 S". In: *Biometrische Zeitschrift* 17.7 (1975), pages 468–468. ISSN: 1521-4037. DOI: 10.1002/bimj.19750170714.

[2] S. A. Andersson, D. Madigan, and M. D. Perlman. "A Characterization of Markov Equivalence Classes for Acyclic Digraphs". In: *The Annals of Statistics* 25.2 (1997), pages 505–541. ISSN: 00905364. URL: http://www.jstor.org/stable/2242556.

[3] D. M. Chickering. "Learning Equivalence Classes of Bayesian-network Structures". In: *J. Mach. Learn. Res.* 2 (Mar. 2002), pages 445–498. ISSN: 1532-4435. DOI: 10.1162/153244302760200696.

[4] D. Colombo and M. H. Maathuis. "Order-independent Constraint-based Causal Structure Learning". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pages 3741–3782. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2750365.

[5] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson. "Learning High-dimensional DAGs with Latent and Selection Variables". In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. UAI'11. Barcelona, Spain: AUAI Press, 2011, pages 850–850. ISBN: 978-0-9749039-7-2. URL: http://dl.acm.org/citation.cfm?id=3020548.3020648.

[6] A. P. Dawid. "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pages 1–31. ISSN: 00359246. URL: http://www.jstor.org/stable/2984718.

[7] M. Kalisch and P. Bühlmann. "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm". In: *J. Mach. Learn. Res.* 8 (May 2007), pages 613–636. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1248659.1248681.

[8] S. L. Lauritzen. *Graphical models*. Volume 17. Clarendon Press, 1996.

[9] T. Le, T. Hoang, J. Li, L. Liu, H. Liu, and S. Hu. "A fast PC algorithm for high dimensional causal discovery with multi-core PCs". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (Feb. 2015).

[10] E. L. Lehmann and J. P. Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[11] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. "NVIDIA Tesla: A Unified Graphics and Computing Architecture". In: *IEEE Micro* 28.2 (Mar. 2008), pages 39–55. ISSN: 0272-1732. DOI: 10.1109/MM.2008.31.

[12] K. Marazopoulou, R. Ghosh, P. Lade, and D. Jensen. "Causal Discovery for Manufacturing Domains". In: *CoRR* abs/1605.04056 (2016). arXiv: 1605.04056.

[13] J. Nickolls, I. Buck, M. Garland, and K. Skadron. "Scalable Parallel Programming with CUDA". In: SIGGRAPH '08 (2008), 16:1–16:14. DOI: 10.1145/1401132.1401152.

[14] NVIDIA Corporation. *CUDA Math API*. July 2017. URL: http://docs.nvidia.com/cuda/pdf/CUDA_Math_API.pdf.

[15] NVIDIA Corporation. *NVIDIA Tesla K80 The World's fastest GPU Accelerator*. Nov. 2014.

[16] NVIDIA Corporation. *Profiler User's Guide*. Mar. 2018. URL: http://docs.nvidia.com/cuda/pdf/CUDA_Profiler_Users_Guide.pdf.

[17] J. Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 978-0-521-89560-6.

[18] R. Penrose. "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Volume 51. 3. Cambridge University Press. Cambridge University Press, 1955, pages 406–413. DOI: 10.1017/S0305004100030401.

[19] A. Rau, F. Jaffrézic, and G. Nuel. "Joint estimation of causal effects from observational and intervention gene expression data". In: *BMC systems biology* 7.1 (Oct. 2013), page 111. DOI: 10.1186/1752-0509-7-111.

[20] T. Richardson. "A Discovery Algorithm for Directed Cyclic Graphs". In: *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. UAI'96. Portland, OR: Morgan Kaufmann Publishers Inc., 1996, pages 454–461. ISBN: 1-55860-412-X. URL: http://dl.acm.org/citation.cfm?id=2074284.2074338.

[21] C. Schmidt, J. Huegle, and M. Uflacker. "Order-independent Constraint-based Causal Structure Learning for Gaussian Distribution Models Using GPUs". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. SSDBM '18. Bozen-Bolzano, Italy: ACM, 2018, 19:1–19:10. ISBN: 978-1-4503-6505-5. DOI: 10.1145/3221269.3221292.

[22] P. Spirtes. "Introduction to Causal Inference". In: *J. Mach. Learn. Res.* 11 (Aug. 2010), pages 1643–1662. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1859905.

[23] P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2000.

[24] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley Publishing, 2009. ISBN: 978-0-470-74366-9.

Exploring Game-Theoretic Formation of Realistic Networks

Tobias Friedrich, Pascal Lenzner, and Christopher Weyand

Algorithm Engineering Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

Many real world networks from different domains share the same structural properties. So far, there only exist models which reproduce these properties via a random process and thus have only limited explanatory power. In contrast to this, we have developed an agent-based game-theoretic model which promises a better explanation of the structure of real world networks. Our new model was investigated in computationally demanding large-scale experiments performed on the hardware of the HPI Future SOC Lab.

1 Introduction

Complex networks from the Internet to various (online) social networks have a huge impact on our lives and it is thus an important research challenge to understand these networks and the forces that shape them. The emergence of the Internet has kindled the interdisciplinary field of Network Science [3], which is devoted to analyzing and understanding real-world networks.

Extensive research, e.g. [1, 3, 4, 6, 14, 17, 18], on real world networks from many different domains like communication networks, social networks, protein-protein interaction networks, neural networks, etc. has revealed the astonishing fact that most of these networks share the following basic properties:

- *Small-world property:* The diameter and average distances in these networks are logarithmic in the number of nodes or even smaller.
- *Clustering:* Two nodes which are both adjacent to a third node have a high probability of being neighbors themselves, i.e. real networks contain an abundance of triangles and small cliques.
- *Power-law degree distribution:* In real networks the probability that a node has degree k is proportional to $k^{-\beta}$, for some constant $2 \leq \beta \leq 5$. That is, the degree distribution follows a power-law. Such networks are called *scale-free networks*.

The phenomenon that real world networks from different domains are very similar calls for a scientific explanation, i.e. formal models which generate networks with the above properties from very simple rules.

Many such models have been proposed, most prominently the preferential attachment model [4], the Chung-Lu random graph model [8], hyperbolic random graphs [13, 15] and geometric inhomogeneous random graphs [5]. However, all these models describe a purely random process which eventually outputs a network having realistic properties. On the one hand, this is desirable for sampling such networks, e.g. for testing algorithms on them, but on the other hand having a purely random process yields only a limited explanation of the structure of real world networks. Most real world networks evolved over time by the interaction of various rational agents. In case of the Internet the selfish agents correspond to the Internet Service Providers which control the Autonomous Systems, in case of social networks, the agents are people or companies who choose carefully with whom to connect. Thus, a model with higher explanatory power should consider rational selfish agents which use and modify the network to their advantage. Such models are at the core of the young field of Algorithmic Game Theory [19, 20].

2 Networks via Game Theory

In game-theoretic models for network formation selfish agents are associated to nodes of a network. Each agent chooses as strategy any subset of other agents to form a link to. The union of all links which are chosen by some player then determines the links of the created network.

The individual goal of each agent is modeled via a cost function, which typically consists of costs for creating links and of a service cost term, which measures the perceived quality of the created network for the individual agent, e.g. the service cost could be the sum of distances to all other agents [12] or just the number of reachable agents [2].

Any assignment of strategies to agents is considered an outcome of the game. Among all those outcomes the so-called equilibria are particularly interesting. In an equilibrium no agent wants to change her current strategy, given that all other players' strategies are fixed, i.e. no agent can reduce her costs in the current situation by forming another set of links. Analyzing the structure of such equilibrium networks then ideally yields insights into why real world networks exhibit the mentioned properties.

So far, such game-theoretic approaches can explain the small-world property, that is, it was proven that the diameter of all equilibrium networks is small [11]. However, to the best of our knowledge, no known game-theoretic model can explain the emergence of clustering and a power-law degree distribution. Thus, it is still an open problem to find and validate such a model.

3 Aims of the Project

Building on our previous work [7, 10, 16], we have developed a new game-theoretic model, called *strategic network augmentation*, which promises to solve the open problem. Initial experiments performed on the Future SOC Lab compute cluster [21] revealed that the obtained equilibrium networks from our new model have the small-world property, show significant clustering and the node degree distribution seems to be governed by a power-law.

This project had two main goals. First, we wanted to verify whether the observed properties for medium size networks can still be found in large scale networks. For this we significantly improved our simulation framework and optimized our generation algorithms. However, even with these improvements it was computationally too demanding to investigate the full parameter space of our model. Hence, we focussed on specific parameter settings which have been found to generate the most promising networks and investigated these variants in depth. The second goal was to investigate a variant of our model which uses only local information. In our original model each agent needs a complete view on the network to compute its perceived service cost. To address this weakness, we replaced the service cost function by a variant which needs only local information, i.e. the size of the 2-neighborhood of the respective agent.

4 Used Future SOC Lab Resources

The experiments were run on the high-performance cluster of the HPI Future SOC Lab. The cluster consisted of 22 nodes with 80 cores each and 1 TB of memory each.[1] The experiments were run via the slurm job scheduler on all nodes in parallel.

In total we generated over 10 000 networks with 1000 to 100 000 nodes and extracted various properties along the way to monitor the process. Later, we analyzed the generated networks using various metrics.

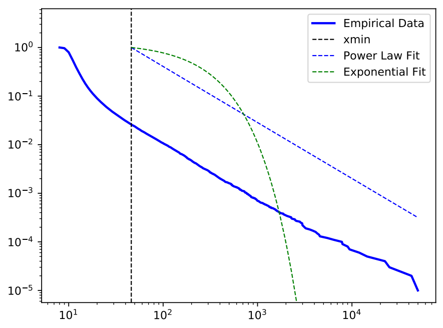
5 Findings

The performed empirical study revealed that our model generates the desired properties of high clustering and a power-law degree distribution (see figure 1a) also for large scale networks.

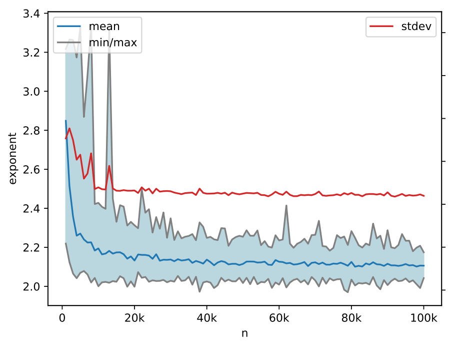
Interestingly, the transition from medium size to large scale networks revealed that a certain networks size is needed to reliably generate a power-law degree distribution for some fixed exponent (see figure 1b). This is especially interesting since this shows that despite the involved randomness in the generation process like the random initial network or the random activation order of the agents a specific parameter setting yields a tightly concentrated power-law exponent.

Concerning our second goal, our experiments revealed that the local variant of our model indeed yields the desired properties, i.e. the generated networks are very similar to the networks generated by the original model. For example, we also get a power-law degree distribution (see figure 1c) and high clustering.

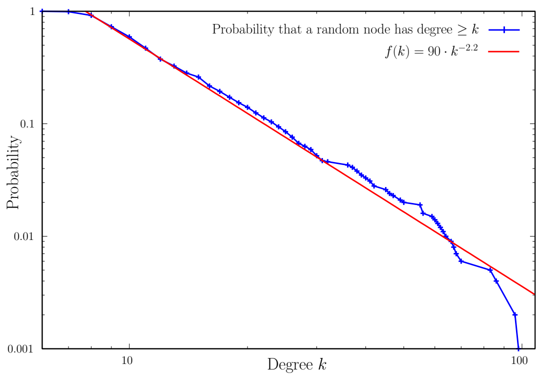
[1]Further hardware specifics can be found here: https://hpi.de/en/research/future-soc-lab/equipment.html



(**a**) Cumulative complementary degree distribution of a generated sample network with 100 000 nodes for our original model. The fitted power-law function has an exponent of $\beta = 2.15$.



(**b**) Measured power-law exponent for networks from 1000 to 100 000 nodes



(**c**) Cumulative complementary degree distribution of a generated sample network with 50 000 nodes for our local model.

Figure 1: A selection of the results of the performed empirical study

However, the experiments with the local version also showed that the generation process is much more time consuming compared with the original version. This is counter-intuitive, since each step in the process is much faster but it turns out that the local version needs significantly more steps until convergence.

6 Next Steps

Our experiments revealed that we need a certain minimum network size to get reliable results. This calls for a systematic re-investigation of the relationship between specific parameter settings and obtained network properties.

We have also observed other realistic properties of the equilibrium networks generated by strategic network augmentation. These need to be further investigated. One example of an additional realistic property is the appearance of a so-called *rich club* [9], which means that the high degree nodes form a connected subgraph. Another example is that our generated networks seem to be resilient against node or edge failure, also commonly observed in real networks.

Last but not least, we want to validate our models with data from real networks. That is, we want to measure if real networks are close to being in equilibrium in our setting and we want to use time series of real networks to investigate if our models predict the right structural changes over time.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. “Internet: Diameter of the world-wide web”. In: *nature* 401.6749 (1999), page 130.

[2] V. Bala and S. Goyal. “A noncooperative model of network formation”. In: *Econometrica* 68.5 (2000), pages 1181–1229.

[3] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[4] A.-L. Barabási and R. Albert. “Emergence of Scaling in Random Networks”. In: *Science* 286.5439 (1999), pages 509–512.

[5] K. Bringmann, R. Keusch, and J. Lengler. “Geometric inhomogeneous random graphs”. In: *arXiv preprint arXiv:1511.00576* (2015).

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. “Graph structure in the Web”. In: *Computer Networks* 33.1 (2000), pages 309–320.

[7] A. Chauhan, P. Lenzner, A. Melnichenko, and L. Molitor. “Selfish Network Creation with Non-Uniform Edge Cost”. In: *SAGT'17*. Springer. 2017, pages 160–172.

[8] F. Chung and L. Lu. “The average distances in random graphs with given expected degrees”. In: *Proceedings of the National Academy of Sciences* 99.25 (2002), pages 15879–15882.

[9] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. "Detecting rich-club ordering in complex networks". In: *Nature physics* 2.2 (2006), page 110.

[10] A. Cord-Landwehr and P. Lenzner. "Network Creation Games: Think Global - Act Local". In: *MFCS'15*. Edited by G. F. Italiano, G. Pighizzini, and D. Sannella. Volume 9235. Lecture Notes in Computer Science. Springer, 2015, pages 248–260. ISBN: 978-3-662-48053-3. DOI: 10.1007/978-3-662-48054-0.

[11] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. "The Price of Anarchy in Network Creation Games". In: *ACM Transactions on Algorithms* 8.2 (2012), page 13.

[12] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. "On a Network Creation Game". In: PODC'03. Boston, Massachusetts: ACM, 2003, pages 347–351.

[13] T. Friedrich and A. Krohmer. "On the diameter of hyperbolic random graphs". In: *ICALP'15*. Springer. 2015, pages 614–625.

[14] J. Kleinberg. "The Small-world Phenomenon: An Algorithmic Perspective". In: *STOC'00*. STOC '00. Portland, Oregon, USA: ACM, 2000, pages 163–170. ISBN: 1-58113-184-4.

[15] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. "Hyperbolic geometry of complex networks". In: *Phys. Rev. E* 82 (3 Sept. 2010), page 036106.

[16] P. Lenzner. "Greedy Selfish Network Creation". In: *WINE'12*. Edited by P. W. Goldberg. Volume 7695. Lecture Notes in Computer Science. Springer, 2012, pages 142–155. ISBN: 978-3-642-35310-9. DOI: 10.1007/978-3-642-35311-6. URL: https://doi.org/10.1007/978-3-642-35311-6.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *SIGKDD'05*. ACM. 2005, pages 177–187.

[18] M. Newman, A.-L. Barabasi, and D. J. Watts. *The structure and dynamics of networks*. Princeton University Press, 2011.

[19] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[20] C. H. Papadimitriou. "Algorithms, games, and the internet". In: *STOC'01*. Edited by J. S. Vitter, P. G. Spirakis, and M. Yannakakis. ACM, 2001, pages 749–753. ISBN: 1-58113-349-9.

[21] D. N. Schumann. "Exploring Game Theoretic Models for Generating Real World Networks". Master's thesis. Hasso Plattner Institute, University of Potsdam, 2018.

PatientSim
Optimizing Patient Similarity Analysis (II)

Lena Wiese[1], Nicole Sarna[2], Ingmar Wiese[3], and Araek Tashkandi[4]

[1] Research Group Knowledge Engineering
Institut für Informatik
Georg-August-Universität Göttingen
wiese@cs.uni-goettingen.de
[2] n.sarna@stud.uni-goettingen.de
[3] ingmar.wiese@stud.uni-goettingen.de
[4] araek.tashkandi@cs.uni-goettingen.de

We extend our investigation of similarity analysis of biomedical data sets. In addition to the previously studied cosine similarity, we also analyze the two additional distance measures Manhattan distance and Euclidean distance. Here we focus on a comparison of their runtime performance. Performance tests were carried out inside a HANA database instance as well as in a distributed VM environment with a machine learning toolkit.

1 Introduction

In this extension of the project, we delve more into the area of efficient computation of patient similarity: that is, for a given target patient a doctor wants to find a group of patients (a so-called "cohort") having similar features (like for example, displaying similar symptoms or having similar lab measurements). By focusing on the cohort, a doctor can decide on an appropriate treatment based on the identified prior experiences or predict future health conditions (including mortality prediction) of the target patient. This patient similarity analysis is a major requirement for personalized medicine [12]. In general, identifying similar cohorts from electronic health records (EHRs) has been conjectured to result in a better overview of available treatment options and health predictions than the usual short-term clinical trials [11] as well as shed more light on consequences of co- and multimorbidities [2]. However, efficiently identifying cohorts with similar health conditions for a given target patient in large and diverse data sets is a major open challenge in personalized medicine.

2 Related Work

Related work on patient similarity [3, 4, 6, 13] has shown that efficiency is still a major issue. So far, similarity computations on medical data have only been done as small proof-of-concept experiments using high-level languages. For example, [11] concede that "[...] Jaccard, Dice and cosine similarities were slow [...] as they were implemented off-the-shelf and had not been optimized for speed". Only few related works analyze the optimization potential offered by modern hardware (like for example [10]). These few existing hardware-optimized implementations for the evaluation of biomedical data cover mostly small-scale fixed data sets. In our own prior work [17] we have already found out that columnar storage of patient data is advantageous.

[7] conclude that an individual patient similarity metric outperforms other statistical models that are based on patient averages. This conclusion is based on their custom developed 30-day-mortality prediction models which use cosine similarity to find the most similar patients. This improvement comes at a higher cost than traditional scores though, namely an increase in computational complexity and load. Their patient data was obtained from the MIMIC-II [8, 14, 15] database which is publicly available for researchers. It contains deidentified information that was collected from almost 30.000 ICU admissions and consist of clinical data such as vital signs every 10-15 minutes, daily lab results, and hourly urine output but also ICD-9 codes, out-of-hospital mortality data, and more. For each patient maximum and minimum values of vital signs in 6 hour periods where extracted, as well as maximum and minimum values of lab results, which are called continuous features and values like the ICD-9 code, called categorical features (cf. [7] for more details). These values constitute the patient vectors that are used to calculate the cosine similarity between to patients:

$$CosineSimilarity = \cos(\theta) = \frac{P_1 \cdot P_2}{\|P_1\| \, \|P_2\|},$$

where θ is the angle between two patient vectors, $P_1 \neq P_2$. Since the cosine of a given angle is always between -1 and 1, two patients are very similar if the result is 1 and very different if the result yields -1. [7] normalized all components of the patient vectors (vital signs etc.) to fit this range in order for them to equally contribute to the patient similarity score. All their calculations were done with *R* (version 3.1.1) while the data was obtained from a *PostgreSQL* database.

3 The Diabetes Dataset

As a test dataset we used the diabetes dataset from the Health Facts database [9, 16]. The dataset was originally extracted from the national Health Facts database (Cerner Corporation, Kansas City, MO). The database contains 10 years (1999–2008) data from EMRs of 130 hospitals throughout the United States. Different data are collected for emergency, outpatient, and inpatient visits—for instance demographic data, ICD code for diagnoses and the procedures, pharmacy data and in-hospital mortality. It consists of 74 036 643 unique visits that correspond to 17 880 231 unique patients with 117 features. The data became available for the researchers after complying with HIPAA by de-identifying all the data. It is available online by the UCI Machine Learning Repository [9]. It is called diabetes dataset since it was extracted to be used for studying the relationship between the measurement of Hemoglobin A1c (HbA1c) and early hospital readmission. Our test dataset was extracted from the original database for inpatients with any kind of diabetes as a diagnosis. Some criteria were considered for the length of stay, laboratory tests and the medications. 101 766 patients fulfill these inclusion criteria.

3.1 Distance Measures

Three distance measures implemented by the ELKI framework were chosen for our analysis.

Manhattan and **Euclidean** of the Minkowski family are widely used distances for continuous datasets [1]. The Euclidean Distance describes the shortest distance between two data points in one line and is calculated as equation (1).

$$d_{Euc} = \sqrt[2]{\sum_{i=1}^{d} |P_i - Q_i|^2} \tag{1}$$

The Manhattan Distance counts the minimum number of units which must be traversed from one point to the other (as shown in equation (2)) and calculates greater distances than Euclidean [5].

$$d_{Manh} = \sum_{i=1}^{d} |P_i - Q_i| \tag{2}$$

The **Cosine** similarity measures (as shown in equation (3)) the cosine of the angle between two vectors [5]. The smaller the calculated angle the closer two data points are to each other. This means if two data points are similar to each other, the angle between them is small and hence the cosine value is near 1.

$$sim_{cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2}\sqrt{\sum_{i=1}^{d} Q_i^2}} \tag{3}$$

In order to turn the similarity into a distance, the ELKI machine learning framework subtracts the cosine similarity value from 1 (as shown in equation (4)).

$$d_{cos} = 1 - sim_{cos} \tag{4}$$

These three distance measures were comparatively evaluated in the HANA database system as well as with the ELKI machine learning framework.

4 Patient Similarity in HANA

In HANA the data set was imported into a column table. We computed the cosine similarity as well as the Manhattan distance and the Euclidean distance in SQL.

For the **cosine similarity**, the norm $\|P\|$ for each patient P was calculated beforehand and stored in an additional column. The cosine similarity was run as a SQL code as shown in listing 1.

Listing 1: SQL code for Cosine similarity

```
SELECT p1.id, p2.id
       (p1.pid_1 * p2.pid_1 +
        ...
        p1.pid_m * p2.pi_m)
       / (p1.norm *p2.norm)
FROM   patients p1
       JOIN patients p2
       ON p1.id < p2.id;
```

For the **Manhattan distance**, the ABS function was used to obtain the absolute value of the difference of two feature values. The Manhattan distance was run as a SQL code as shown in listing 2.

Listing 2: SQL code for Manhattan distance

```
SELECT p1.id, p2.id,
       ABS(p1.pid_1 - p2.pid_1)+
       ...
       ABS(p1.pid_m - p2.pid_m)
FROM   patients p1
       JOIN patients p2
       ON p1.id < p2.id;
```

For the **Euclidean distance**, the POWER function was used to obtain the exponentiation of the difference of two feature values. The Euclidean distance was run as a SQL code as shown in listing 3.

Listing 3: SQL code for Euclidean distance

```
SELECT p1.id, p2.id, sqrt (
       POWER(p1.pid_1 - p2.pid_1,2)+
       ...
       POWER(p1.pid_m - p2.pid_m,2))
FROM   patients p1
       JOIN patients p2
       ON p1.id < p2.id;
```

We applied batch processing in the sense that the similarity for 50 patients at the same time was calculated. The runtime per individual patient is shown below. The overall computation of all pairwise similarities took roughly 36 hours in total which leaves room for optimization. However, in order to have a qualitative comparison of the three measures, we sampled a random 10 % of the data set. The comparison of the three measures on the sample is shown in figure 1.

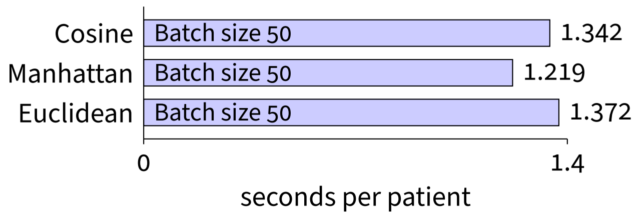


Figure 1: Hana on a sample of 10 % the dataset

5 Patient Similarity in VMs

Using the three provided FSOC VMs, the three distances among the patients of the diabetes dataset were calculated. To facilitate the computation of the distances in Java, we again utilized the jars of the Environment for Developing KDD-Applications Supported by Index-Structures (ELKI)[5] developed by the chair Database and Information Systems of the Institute for Computer Science at the University of Munich. Since the VMs had no Java version pre-installed, we decided to locally set the VMs shells to use the latest Java JDK from Oracle. Further, we used Apache's Commons CSV[6] to effectively iterate over each row of the accomplished CSV file of the diabetes dataset.

During the iteration we created for each recorded admission a patient vector regardless if a patient occurred more frequently in the dataset. The dimension of a patient vector is three times greater than the amount of features in the dataset, due to factorization of binary features like the gender, or features containing a specific amount of different ids like the admission type with its nine distinct values. To enable the assignment of the vectors to a specific patient, each vector was put as value and its unique admission id as key in a `Map<String, Vector>`.

After vectorizing the patients, we calculated the three distances and printed the results for each distance in its own file using the schema ⟨patientID-1, patientID-2, distance⟩. To calculate the distances between each patient vector with another efficiently, we firstly decided to let the three VMs calculate the vector distances in

[5]https://elki-project.github.io/ (last accessed 2018-10-13).

[6]https://commons.apache.org/proper/commons-csv/ (last accessed 2018-03-15).

parallel (see figure 2) with the result that VM1 executed the distance computation between each vector of the first half of the vector map, VM2 did the same for the second half, and VM3 calculated the distances between the vectors of the first half and the vectors of the second half. Secondly, we never calculated the distance between a patient vector with itself and only executed the distance calculation for two vectors one-sided, visualized in figure 2 as white background, meaning if a we already have the distance between patientID-1 and patientID-2, we will not need to calculate it for patientID-2 and patientID-1, since it will lead to the same distance.

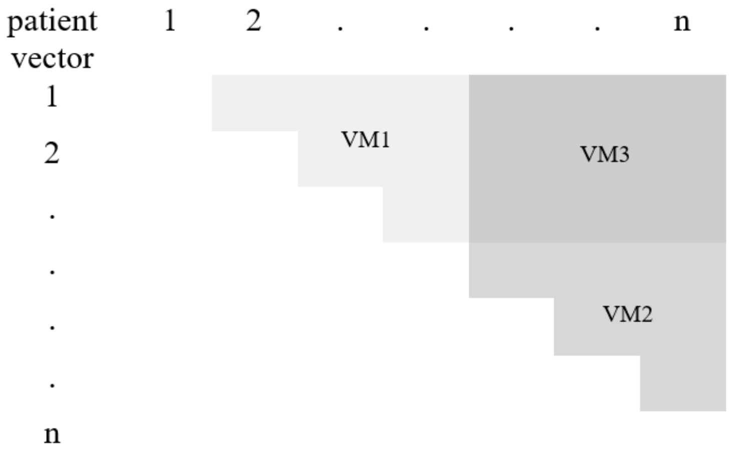


Figure 2: VMs Vector Calculation Matrix

We created a separate output file for each VM containing the calculated distances between the several patients. The time for calculating and writing each file was measured. The results are shown in figure 3 to figure 5.

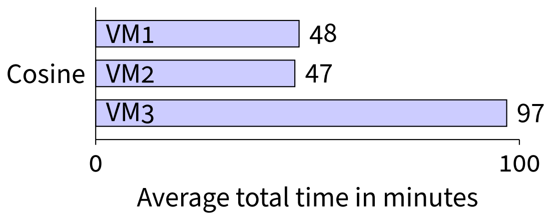


Figure 3: VMs Performances for Cosine

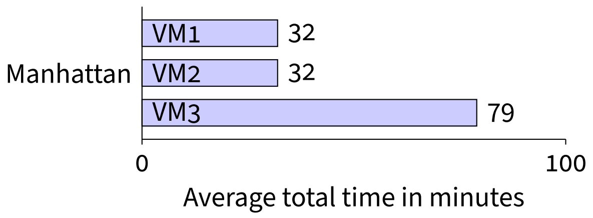


Figure 4: VMs Performances for Manhattan

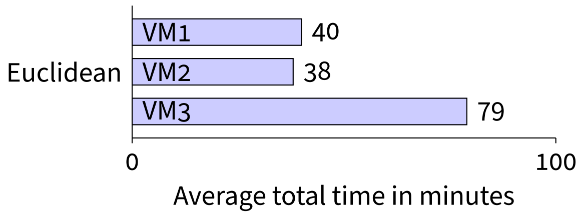


Figure 5: VMs Performances for Euclidean

6 Conclusion and Future Work

Computing the distances for all pairs of patients in our data set was currently significantly faster with the machine learning tool on the three VMs than with the single HANA instance. We can conclude from all tests that the Manhattan distance is the one with the best runtime behavior. Further investigation with stratified 10-fold cross-validation for *k*-nearest neighbor search showed that the Manhattan distance was also best in terms of accuracy. For future work, several other distance measures as well as prediction methods can be investigated in depth.

References

[1] V. Chandola, S. Boriah, and V. Kumar. *Similarity Measures for Categorical Data–A Comparative Study*. Technical report 07-022. University of Minnesota, Department of Computer Science and Engineering, Oct. 15, 2007. HDL: 11299/215736.

[2] B. Gallego, S. R. Walter, R. O. Day, A. G. Dunn, V. Sivaraman, N. Shah, C. A. Longhurst, and E. Coiera. "Bringing cohort studies to the bedside: framework for a 'green button'to support clinical decision-making". In: *Journal of comparative effectiveness research* 4.3 (2015), pages 191–197.

[3] D. Girardi, S. Wartner, G. Halmerbauer, M. Ehrenmüller, H. Kosorus, and S. Dreiseitl. "Using concept hierarchies to improve calculation of patient similarity". In: *Journal of biomedical informatics* 63 (2016), pages 66–73.

[4] A. Gottlieb, G. Y. Stein, E. Ruppin, R. B. Altman, and R. Sharan. "A method for inferring medical diagnoses from patient similarities". In: *BMC medicine* 11.1 (2013), page 194.

[5] J. W. Johnston. *Similarity Indices I: What Do They Measure?* Technical report. Battelle Pacific Northwest Labs., Richland, WA (USA), 1976.

[6] J. Lee, D. M. Maslove, and J. A. Dubin. "Personalized mortality prediction driven by electronic medical data and a patient similarity metric". In: *PloS one* 10.5 (2015), e0127428.

[7] J. Lee, D. M. Maslove, and J. A. Dubin. "Personalized mortality prediction driven by electronic medical data and a patient similarity metric". In: *PloS one* 10.5 (2015), e0127428.

[8] J. Lee, D. J. Scott, M. Villarroel, G. D. Clifford, M. Saeed, and R. G. Mark. "Open-access MIMIC-II database for intensive care research". In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE. 2011, pages 8315–8318.

[9] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[10] Y. Lou, A. Irimia, P. A. Vela, M. C. Chambers, J. D. Van Horn, P. M. Vespa, and A. R. Tannenbaum. "Multimodal deformable registration of traumatic brain injury MR volumes via the Bhattacharyya distance". In: *IEEE Transactions on Biomedical Engineering* 60.9 (2013), pages 2511–2520.

[11] Y. S. Low, B. Gallego, and N. H. Shah. "Comparing high-dimensional confounder control methods for rapid cohort studies from electronic health records". In: *Journal of comparative effectiveness research* 5.2 (2016), pages 179–192.

[12] J. L. Mega, M. S. Sabatine, and E. M. Antman. "Population and personalized medicine in the modern era". In: *JAMA* 312.19 (2014), pages 1969–1970.

[13] M. Panahiazar, V. Taslimitehrani, N. L. Pereira, and J. Pathak. "Using EHRs for heart failure therapy recommendation using multidimensional patient similarity analytics". In: *Studies in health technology and informatics* 210 (2015), page 369.

[14] M. Saeed, C. Lieu, G. Raber, and R. G. Mark. "MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring". In: *Computers in Cardiology*. IEEE. 2002, pages 641–644.

[15] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark. "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): a public-access intensive care unit database". In: *Critical care medicine* 39.5 (2011), page 952.

[16] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, and J. N. Clore. "Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records". In: *BioMed research international* 2014 (2014).

[17] A. Tashkandi and L. Wiese. "Leveraging Patient Similarity Analytics in Personalized Medical Decision Support System". In: *LWDA*. 2017.

CIG: Cyber Intelligence Graph
Big Data Analytics for Security

Pejman Najafi, Mirko Krause, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute
University of Potsdam
{pejman.najafi,feng.cheng,christoph.meinel}@hpi.de
Mirko.Krause@student.hpi.uni-potsdam.de

While the majority of today's enterprises store valuable security-related data within their SIEM systems in the traditional manner, i.e. a flat data structure for time series data. In this research we investigate the usage of knowledge graphs in security by; first, creating an ontology for SIEM related data as well as related open source intelligence (OSINT). Second, designing and implementing a platform which is capable of ingesting raw data into a scalable graph database.

1 Introduction

Nowadays, the majority of organizations, collect and store valuable event logs and telemetry generated by different components in the organization's premises, e.g. proxy servers, DNS servers, firewalls, workstations, etc. This is usually achieved by utilizing a Security Information and Event Management (SIEM) solution. Investigating the data captured within SIEM systems pose the potential to uncover security threats.

Traditionally, a Security Operations Center (SOC) analyst is tasked to monitor the SIEM dashboard for anomalies or triggered alerts. Moreover, to query and investigate those possible issues which involves joining data across various sources. This task is extremely challenging and unintuitive due to the variety and volume of data.

In this research we investigate the usage of knowledge graphs in security by; first, creating an ontology for SIEM related data as well as related open source intelligence (OSINT). This graph model allows complex structures and semantics, representing various TI and OSINT information together with SIEM related data. Second, designing and implementing a platform which is capable of ingesting raw data into a scalable graph database (CIG).

The SIEM system within organizations is expected to be the centralized repository for all relevant security-related data. That being said, the amount and variety of the data ingested into SIEM systems vary drastically depending on the organization and its dedication to security practices, compliances, and analytics. In this regard, it is important to define a scope for our initial expectation on SIEM-related data.

For the purpose of this research, we position ourselves in the place of a SIEM system located in a large international organization which collects various event logs, more specifically proxy, DNS, firewall, and DHCP.

OSINT refers to information gathered from open sources (i.e. information available to the general public). OSINT can provide the context to those observed entities extracted from SIEM event logs (e.g. IP ranges, X.509 certificates, whois data, and etc.). Threat intelligence is information that can aid with threat detection tasks, also known as indicators of compromise (e.g. list of malicious domains and IPS, file hashes, and etc.). Similar to event logs, OSINT and TI data could also pivot endlessly. OSINT Framework[1] and Enaqx[2] provides a good overview of all available OSINT sources, and Slatman[3] provides a curated list of threat intelligence. For this research we are going to limit the usage of OSINT/TI t those data which is related to our SIEM. Figure 1 shows CIG full graph schema.

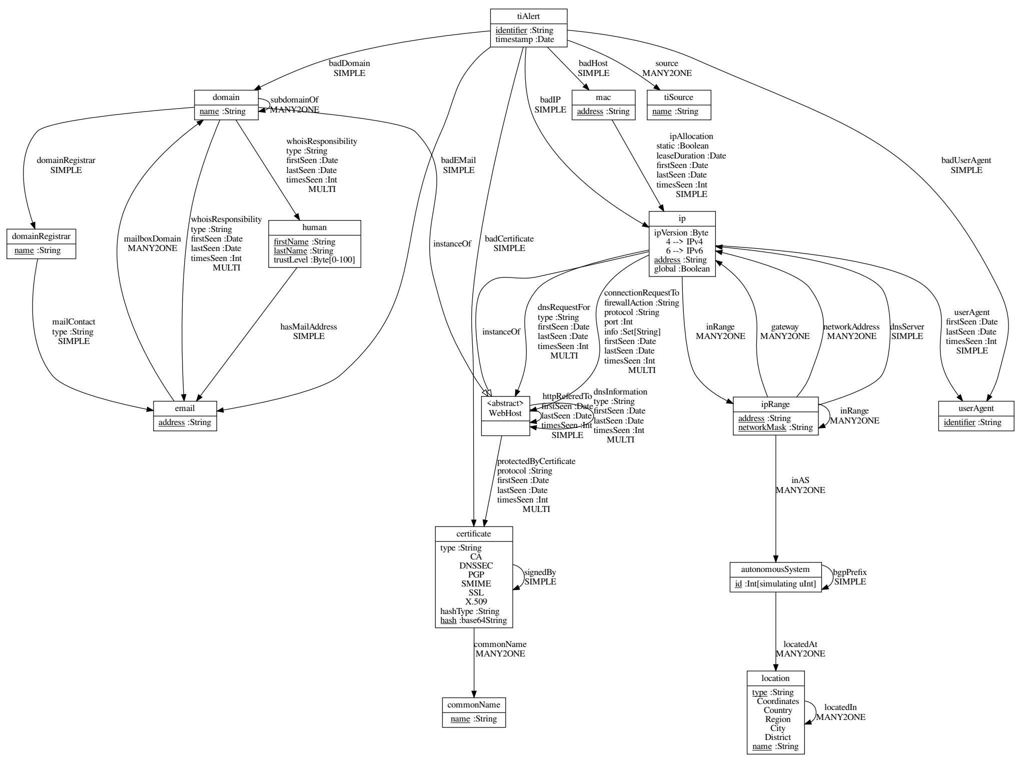

tiAlert
identifier :String
timestamp :Date
badDomain SIMPLE
badHost SIMPLE
source MANY2ONE
badIP SIMPLE
badEMail SIMPLE
badCertificate SIMPLE
badUserAgent SIMPLE
domain
name :String
subdomainOf MANY2ONE
mac
address :String
tiSource
name :String
domainRegistrar SIMPLE
domainRegistrar
name :String
whoisResponsibility type :String firstSeen :Date lastSeen :Date timesSeen :Int MULTI
human
firstName :String
lastName :String
trustLevel :Byte[0-100]
mailboxDomain MANY2ONE
whoisResponsibility type :String firstSeen :Date lastSeen :Date timesSeen :Int MULTI
mailContact type :String SIMPLE
hasMailAddress SIMPLE
email
address :String
instanceOf
ipAllocation static :Boolean leaseDuration :Date firstSeen :Date lastSeen :Date timesSeen :Int SIMPLE
ip
ipVersion :Byte
4 --> IPv4
6 --> IPv6
address :String
global :Boolean
instanceOf
dnsRequestFor type :String firstSeen :Date lastSeen :Date timesSeen :Int MULTI
connectionRequestTo firewallAction :String protocol :String port :Int info :Set[String] firstSeen :Date lastSeen :Date timesSeen :Int MULTI
inRange MANY2ONE
gateway MANY2ONE
networkAddress MANY2ONE
dnsServer SIMPLE
userAgent firstSeen :Date lastSeen :Date timesSeen :Int SIMPLE
<abstract> WebHost
httpReferedTo firstSeen :Date lastSeen :Date timesSeen :Int SIMPLE
dnsInformation type :String firstSeen :Date lastSeen :Date timesSeen :Int MULTI
ipRange
address :String
networkMask :String
inRange MANY2ONE
userAgent
identifier :String
protectedByCertificate protocol :String firstSeen :Date lastSeen :Date timesSeen :Int MULTI
inAS MANY2ONE
certificate
type :String CA DNSSEC PGP SMIME SSL X.509
hashType :String
hash :base64String
signedBy SIMPLE
autonomousSystem
id :Int[simulating uInt]
bgpPrefix SIMPLE
commonName MANY2ONE
locatedAt MANY2ONE
commonName
name :String
location
type :String Coordinates Country Region City District
name :String
locatedIn MANY2ONE

Figure 1: Knowledge graph schema, showing the extracted entities and relationship

[1]http://osintframework.com/ (last accessed 2018-04-01).
[2]https://github.com/enaqx/awesome-pentest#osint-tools (last accessed 2018-04-01).
[3]https://github.com/hslatman/awesome-threat-intelligence/ (last accessed 2018-04-01).

2 CIG Design and Implementation

2.1 High-level Architecture

The CIG platform consists of two general parts: the graph backend and the processing engine(s). Figure 2 provide the high level architecture design.

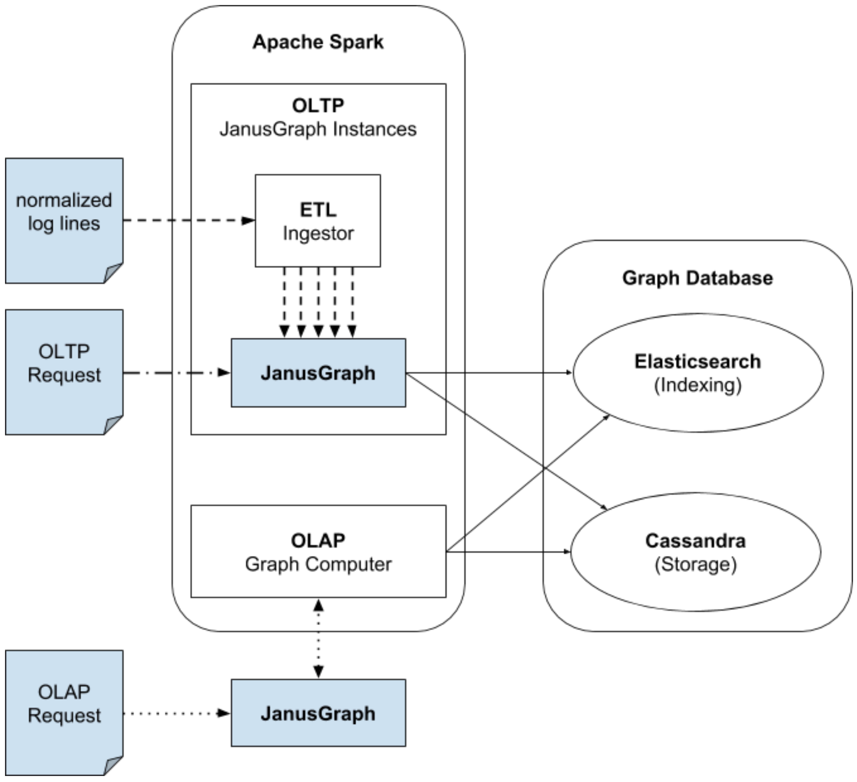


Figure 2: CIG High-level Architecture

Due to our requirements, i.e. scalability, big data, OLTP as well as OLAP support, for this research we decided to utilize JanusGraph [5], the successor of TitanDB. JanusGraph is simply a java library, that gets executed by clients. It connects to a configured storage backend, where the actual data is stored, e.g. Apache Cassandra [1] or Apache HBase [2].

For the processing engine of the CIG platform, Apache Spark [3] was chosen, due to its maturity in big data community for large scale data processing. More specifically, for the OLTP operations the CIG platform utilizes Spark to spawn many JanusGraph instances which in parallel process (ingest) the data.

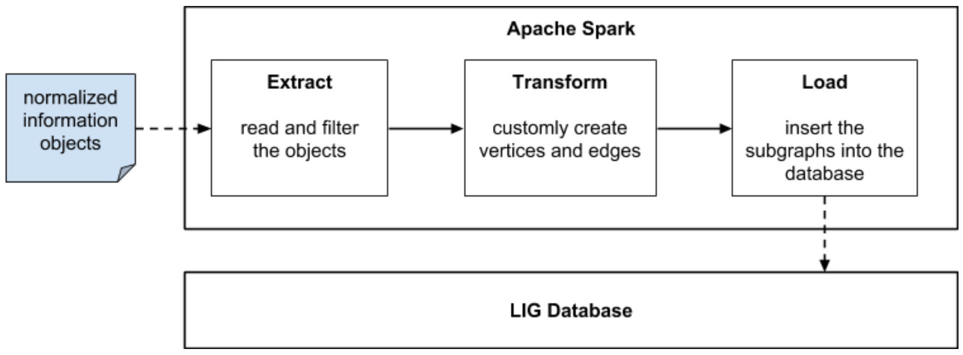

Apache Spark
normalized information objects
Extract
read and filter the objects
Transform
customly create vertices and edges
Load
insert the subgraphs into the database
LIG Database

Figure 3: Data Ingestion Workflow

2.2 High-level Workflow

As shown in figure 3, at high level, the first module is the **extraction**, where entities and relationships of interest are extracted from the raw data presented either though a distributed queue (e.g. Apache Kafka) or file on a distributed file systems (e.g. HDFS). Next, in the **transformation** module, the extracted, parsed, and cleaned data rows are mapped to subgraphs using custom functions for each log type. The subgraphs consist of lists for generic vertices and their connecting edges. Lastly, within the **loading** module. The results (sub graphs) will be inserted into the graph database.

3 Discussions

3.1 Time Series Events

There are different approaches for representing time-dependent data in graphs. More specifically:

First, using a timestamped edge for every log event. While this could be useful since there will be no information loss, it introduces a huge number of duplicated edges, which can cause performance issues in long run.

Second, using a time tree, this can be achieved by building a time tree [4], where year instances are connected to month instances connected to day instances. The lowest level then gets connected to event nodes as described in [6]. This could be even further improved by the usage of hyper edges. In that case the events within the same time interval can efficiently be correlated and the information loss is dependent on the interval size. In addition, trends can be interpolated. However, the graph schema becomes much more complex due to hyperedges. This causes traversals to become inefficient (due to two step traversal). Furthermore, the time nodes can become supernodes, if the time interval is too large.

Third, complete aggregation of similar edges. In this solution, one will only store the timestamps of the first and last seen log event connecting two nodes with the same edge type as well as the count of such seen events. It can be extended by a list storing the single timestamps or aggregating those again per interval like before and storing for each interval the seen count. In this solution the graph schema will be kept simple and easier to implement. Also duplication is at its minimum and at least time ranges are kept. On the other hand, all concrete time information gets lost, so it is no longer useful for finding trends and continuous behavior as long as the extension is not implemented.

3.2 Supernodes

Supernodes are nodes with large number of edges which therefore are inefficient to traverse over (e.g. google.com).

The naive solution tackling the supernodes is to blacklist them all. Alternatively, one could reduce time-to-life (TTL) for vertices and edges, so that old information gets deleted by the database if not needed anymore. This is still no final solution, because aggressive supernodes could require the TTL to become shorter until there is no useful data left.

4 Evaluation

4.1 Hardware Setup

For the purpose of this research three servers connected by a 10 Gbit/s network were utilized. More specifically, each having:

- 16 cores, 32 threads @ 2.4 GHz
- 157 GB DDR4 RAM @ 2133 MHz
- 1 HDD @ 10k RPM

4.2 Ingestion Experiment

For the first experiment, a small portion of our data was used to test the performance of CIG implementation in ingesting the related data. JanusGraph was configured with simple indexing directly on the Cassandra site. Note that the majority of those indexes were unique indexes. In this regard, If an index has uniqueness checks enabled, locking on Cassandra site is needed to keep the index consistent. Table 1 shows the details of this experiment.

4.3 Performance Investigation

According to our initial investigation the platform did not perform as expected, particularly considering its underlying hardware. Looking at the CPU utilization while data is actively written to the database, even 96 parallel threads, for this setup,

Table 1: First experimental usage of CIG with Proxy and DHCP logs

log type	DHCP	Web Proxy Th
data format	JSON	JSON
file size	1.7 GB	3.1 GB
# of log events	3 143 136	1 161 306
time frame	11 154 s ~ 186 min ~ 3 h	607 s ~ 10 min
applied filtering	only DHCPACK events	-
# of used log events	898 577	1 161 306
# of created vertices	all: 163 679 mac: 79 222 ip: 84 457	all: 74 509 domain: 25 231 userAgent: 630 ip: 48 648
# of created edges	ipAllocation: 898 577	all: 3 472 543 connectionRequestTo: 1 161 306 userAgent: 1 161 306 dnsInformation: 1 149 931

do not manage to use the full CPU power. Only 20 percent to 40 percent are used by Spark and Cassandra together (because they are both running on every machine). The network IO is probably faster than the hard drive IO, because, unfortunately, the Cassandra database had to use the slowest installed hard drives, so these with 10k RPM. If their optimistic throughput even is around 250 Mbit/s (taken from a sample hard drive by Seagate [7]), it still only sums up to 2 Gbit/s, one fifth of the network throughput. Their average latency of more than 2 ms is probably above the network latency too.

This means, on such fast servers Cassandra should at least be running on an SSD or a very fast RAID system to at least utilize the whole computing resources.

This is definitely necessary to at least store the data with the same speed as it arrives, even though most real-world event log streams are smaller than the ones that were used for testing. Still even the used ones are taken from the real world.

5 Future Work

One of the main purpose of this research was to investigate the performance and efficiency of a graph based SIEM system. This task turned our to be extremely challenging due to number of variables involved. Within this first attempt we managed to spawn the platform and its underlying technologies on the top of Kubernetes. However we could not finish our detailed investigation. Moreover, we did not investigate yet the performance gain upon queries compared to traditional SQL database. In this regard, one of the main reason for the usage of graph database underneath instead of traditional SQL based DB was to overcome the slow joining process of highly correlated data.

Lastly, we would like to expand our schema to support more sources from both SIEM system and OSINT. These are the main direction we would like to take in the future.

6 Conclusion

In this research, we investigate the use of graph databases in security to create a cyber knowledge graph which can be used by SOC analyst to investigate threats and potential indicators much more efficient and intuitive. We discussed how event logs and OSINT/TI data can be correlated.

In this regards, a graph schema was developed on the basis of a deep analysis of the data sources. The schema is able to combine large sets of security data objects without losing important information. Therefore some aspects of the graph model have been discussed.

We also discussed the underlying design and implementation for such a platform to support our main requirements which were scalability to handle big data, OLTP style transactions, and OLAP style analysis on large graphs.

References

[1] *Apache Cassandra*. URL: https://cassandra.apache.org/ (last accessed 2018-07-05).

[2] *Apache HBase – Apache HBase™ Home*. URL: https://hbase.apache.org/ (last accessed 2018-07-05).

[3] *Apache Spark™ - Unified Analytics Engine for Big Data*. URL: https://spark.apache.org/ (last accessed 2018-07-05).

[4] Graph Aware Limited. *GraphAware Neo4j TimeTree*. GraphAware. URL: https://graphaware.com/neo4j/2014/08/20/graphaware-neo4j-timetree.html (last accessed 2018-07-05).

[5] *JanusGraph: Distributed graph database*. URL: http://janusgraph.org/ (last accessed 2018-07-05).

[6] *Modeling Time Series Data with Neo4j*. GraphGrid. July 6, 2015. URL: https://www.graphgrid.com/modeling-time-series-data-with-neo4j/ (last accessed 2018-07-05).

[7] *Seagate Enterprise Performance 10K HDD Review | StorageReview.com - Storage Reviews*. May 18, 2015. URL: https://www.storagereview.com/seagate_enterprise_performance_10k_hdd_review (last accessed 2018-07-05).

DevOps–oriented Declarative Load Testing for Microservices

André van Hoorn, Vincenzo Ferme, Markus Frank, and Henning Schulz

University of Stuttgart
Institute of Software Technology

This report provides a summary of our project "DevOps-oriented Declarative Load Testing for Microservices" conducted during the HPI Future SOC Lab period spring 2018, as well as ideas for a follow-up project for the upcoming period.

1 Introduction

Modern software engineering paradigms and technologies — such as DevOps [2] (including automation as part of continuous delivery) and microservices [13] — are gaining more and more attraction in the software and services engineering communities. Of particular interest are quality-of-service concerns, for instance, w. r. t. performance and reliability. While established approaches for classic contexts (i.e. which do not use DevOps and microservices) exist, their adoption to DevOps and microservices requires considerable research efforts [3, 11].

In the recent years, our group has already contributed architecture-aware approaches for performance and reliability, involving a combination of measurement-based and model-based techniques [12, 14, 16]. Recently, we started to investigate how these techniques can be used in DevOps and microservice contexts — with a particular focus on load testing as a key performance engineering activity [5, 15, 16].

In order to conduct large-scale experimental evaluations, we need a state-of-the-art computing infrastructure such as the one provided by the HPI Future SOC Lab.

We have requested the HPI Future SOC Lab for setting up an infrastructure for the experimental evaluation of our ongoing research on continuous performance testing for microservices. We have adopted the BenchFlow [4] benchmarking framework to support microservices and used it for extensive exploratory performance testing of microservices based on a novel approach.

In the remainder of this report, we will list the granted Future SOC Lab resources, provide a brief description of our conducted activities as part of the project, and outline next steps. First of all, the activities on load testing conducted during this period, were a direct continuation of the activities started during the previous two periods. Additionally, we conducted experimental evaluations for our ongoing research on software performance engineering for multi-core systems [6, 7, 8, 9].

2 Granted Future SOC Lab Resources

We requested and received dedicated (root) access to the following computing resources (servers): *i.*) 896 GB RAM, 80 cores; *ii.*) 32 GB RAM, 24 cores. Dedicated access has been given to us due to our expected high resource demands.

3 Project

The project goals as stated in the proposal were:

3.1 Declarative Load Testing

We started to use BenchFlow as an execution framework for automating the load test execution. We have executed extensive experimentation to assess BenchFlow's capability in providing reliable and reproducible results on the HPI infrastructure. We plan to integrate BenchFlow also into our attempts on Declarative Performance Engineering [17]. For doing so we are going to extend the declarative language and execution framework provided by BenchFlow, and relying on extensive experimentation to assess the possibility to abstract complexity from the developers and performance engineers when executing complex performance test activities.

3.2 Detection of Performance Regressions based on Load Tests

One use case of load testing is the detection of performance regressions that have been introduced during the development. We plan to develop a declarative approach for microservices, so that developers can specify the regression criteria to be met and detect violation of those during continuous development of software systems. To accomplish so, we plan to rely on state-of-the-art techniques for regression detection and execute extensive experiments to assess the capability of the declarative approach to correctly and automatically report regression issues. As part of this, we plan to experimentally assess the suitability of performance metrics for regression detection of microservices.

3.3 Prioritization and Selection of Load Tests

Load tests are resource-intensive and time-consuming. Hence, it is infeasible to execute the entire test suite for each software change as part of the continuous delivery pipeline. Instead, it is desirable to run only those tests that are (performance-)relevant for the most recent change. We plan to continue our activities for efficient load testing started in the previous period, and investigate additional techniques for prioritizing and selecting load tests. For example, we plan to mine the data collected by BenchFlow about past executions, and perform additional experiments by deploying the services on top of a Kubernetes cluster, with the aim of defining a Key

Performance Indicator (e.g. maximum duration of tests, maximum use of resources, or desired test coverage) enabling us to prioritize the execution of performance tests, when no time is available to execute the entire test suite. An additional input to the algorithms is contextual data, e.g. information about the usage profile expected during deployment of the next version. We then plan to provide a declarative specification to perform this kind of performance analysis, and integrate the possibility to automate the mentioned analysis by relying on the BenchFlow framework.

3.4 Advanced Extraction of Load Test Specifications from APM Data

Specification of a load test that is representative for the actual user behavior is challenging. In our previous work, we developed an approach to generate representative load test specifications from APM data [10] using the WESSBAS approach [16]. As an advancement of this, we plan to take special account of the evolution of the monitored system as well as of microservice architectures. Since the behavior of the users of the target system as well as the system interface can change, we want to semi-automatically detect and merge these changes into already existing load tests. Furthermore, we are going to modularize load test specifications to apply the load test generation approach to single microservices or subsets of all of the target system's microservices.

3.5 Case Studies

As a system under test, we have so far mainly used the Sock Shop developed by Weaveworks. It represents an e-commerce website and is available under an open-source license. In order to increase the generalizability of our results, we plan to experiment with additional microservices-based systems under test. Possible candidates are the Pet Supply Store developed by the University of Würzburg or the next version of the SPECjEnterprise ® industry-standard benchmark developed by the Standard Performance Evaluation Corporation.

4 Description of Activities and Selected Results

We have worked on all previously stated goals. Not for all of them, we have conducted experiments in the HPI Future SOC Lab. Additional experiments are planned for the next period.

A paper including experiments conducted in the HPI Future SOC Lab has been accepted for the 12th European Conference on Software Architecture (ECSA 2018) [1]. We had presented preliminary results at the HPI Future SOC Lab Day in spring 2018. Moreover, we conducted a new type of experiments for our research on software performance engineering for multi-core systems. In this section, we will provide a summary of the experiments and results for these two types of experiments.

4.1 A quantitative approach for the assessment of microservice architecture deployment alternatives using automated performance testing

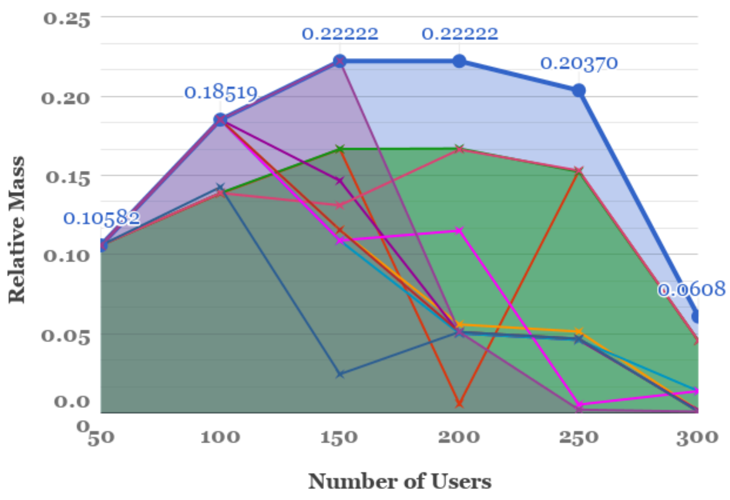


Figure 1: Relative and best test masses per number of users in the HPI environment [1]

In the ECSA 2018 paper [1], we introduce a quantitative approach for the performance assessment of microservice deployment alternatives. The approach uses automated performance testing results and high-level performance modeling to quantitatively assess each architectural configuration in terms of a domain metric introduced in the paper. For performance testing, we focus on load tests based on operational workload situations.

We have evaluated the introduced domain metric for ten different configurations based on two different memory allocations, two different CPU allocations, and three different values for the number of Docker container replicas. The experiments were executed in two data center environments, one if it being the HPI infrastructure. We evaluated for each environment the best performing architectural configuration.

As a system under test (SUT), we have used the Sock Shop developed by Weaveworks.[1] It represents an e-commerce website and is available under an open-source license. The distributed application has been designed and implemented based on the microservices architectural style [13] and respective state-of-the-art technologies. Each microservice is available as a Docker[2] image, which is a container-based virtualization technology. We used BenchFlow as an execution framework for au-

[1]https://microservices-demo.github.io/ (last accessed 2018-04-01).
[2]http://docker.com (last accessed 2018-04-01).

tomating the load test execution of services deployed in Docker containers on top of the Rancher[3] orchestration framework, by relying on the provided Declarative Domain Specific Language for defining performance tests. Previously, we have executed extensive experimentation to assess BenchFlow's capability in providing reliable and reproducible results on the HPI infrastructure.

We deployed the SUT using ten different architectural configurations. The parameters we vary over the different configurations are the amount of available RAM, the CPU share, and the replicas for the cart service. We target the Sock Shop's cart service, as most of the requests issued by the designed workload target the cart service. We set the RAM to [0.5 GB, 1 GB], the CPU share to $[0.25, 0.5]$, and the number of replicas to $[1, 2, 4]$.

Details about the approach and the experimental setup are provided in the publication [1] and in the previous report.

Figure 1 shows the test masses for the different investigated architectural configurations in relation to the workload situations (numbers of users). The best relative test mass plot represents the theoretical maximum which is reached if all tests pass. It can be seen from Figure 1 that none of the alternatives reached the best relative mass, because of scalability assessment failures identified. For the HPI environment (bare metal), the configuration with 1 GB of RAM, 0.5 CPU share, and four cart replicas not have failures for up to 150 users. However, the relative mass decreases significantly when the number of users is increased.

For the HPI (bare metal) experiments, the configuration with 1 GB of RAM, 0.25 CPU share and one cart replica has the highest domain metric $D(\alpha, S) \approx 0.78$, followed by the configuration with 1 GB of RAM, 0.25 CPU share, and two cart replicas having a metric value of about 0.74. The worst configuration with $D(\alpha, S) \approx 0.37$ is 1 GB of RAM, 0.25 CPU share, and four cart replicas. This is an interesting result with significant implications to the assessment of architectural deployment alternatives, since adding additional replicas with the same memory and CPU configuration may decrease the application's performance for the HPI environment.

4.2 Obtaining Performance Curves for Software Performance Engineering for Multi-Core Systems

Multicore systems are a permanent part of our daily life. Regardless of whether we consider nowaday's desktop PCs, notebooks, or smart phones — all devices are running on multicore CPUs. To use these hardware features in an efficient way, developers need to build parallel-enabled software. However, the development of such software is more complex than developing sequential software.

To handle the rising complexity, it is necessary to develop software in an engineering-like way. In such a process software architects plan and analyze software designs on a model level. Software architects can use tools like Palladio to simulate and analyze early-phase software designs. Unfortunately, current approaches and

[3]http://rancher.com (last accessed 2018-04-01).

tools lack the ability to consider multicore systems. Therefore, in this project, we aim to find performance prediction methods for multicore systems.

In this period, we have started to use the HPI infrastructure to study performance properties for performance predictions of multi-core systems. The experiments comprised different configurations (e.g. thread numbers and threas pool sizes) to assess what is their impact on performance properties for different use cases or scenarios (e.g. benchmarks) to obtain performance curves. These performance curves will be used by software architects to easily adopt performance prediction models.

The experiments are in the context of our ongoing research in the area of software performance engineering for multi-core systems [6, 7, 8, 9].

5 Next Steps

For the next HPI Future SOC Lab period, we have proposed the project "Measurement-Based Software Performance Engineering for Microservices and Multi-Core Systems". We divide our planned activities into two subprojects, namely (1) "DevOps-oriented Load Testing for Microservices" and (2) "Software Performance Engineering for Multi-Core Systems". Subproject (1) is a direct continuation of the works that we started in the previous periods. Subproject (2) continuous the initial experiments on the HPI infrastructure from this period.

6 Conclusion

We are thankful to the HPI Future SOC Lab for having granted us access to the computing infrastructure. The environment eases the joint work of different organizations on the platform, which has so far been hindered by university-internal access constraints—apart from the fact that an equipment comparable to that of the HPI Future SOC Lab has not been available to us, and that enables extensive performance configuration tests needed to reach our goals.

Acknowledgment

The presented works are a joint effort being conducted together with many researchers from different organizations, including (in alphabetical order) Alberto Avritzer, Andrea Janes, and Barbara Russo.

References

[1] A. Avritzer, V. Ferme, A. Janes, B. Russo, H. Schulz, and A. van Hoorn. "A quantitative approach for the assessment of microservice architecture deploy-

ment alternatives using automated performance testing". In: *Proceedings of the 12th European Conference on Software Architecture (ECSA 2018)*. LNCS. Springer, 2018.

[2] L. J. Bass, I. M. Weber, and L. Zhu. *DevOps — A Software Architect's Perspective*. SEI series in software engineering. Addison-Wesley, 2015. ISBN: 978-0-13-404984-7.

[3] A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. Herbst, P. Jamshidi, R. Jung, J. von Kistowski, A. Koziolek, J. Kroß, S. Spinner, C. Vögele, J. Walter, and A. Wert. *Performance-oriented DevOps: A Research Agenda*. Technical report SPEC-RG-2015-01. SPEC Research Group — DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), 2015.

[4] V. Ferme and C. Pautasso. "A Declarative Approach for Performance Tests Execution in Continuous Software Development Environments". In: *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18)*. 2018.

[5] V. Ferme and C. Pautasso. "Integrating Faban with Docker for Performance Benchmarking". In: *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016)*. Delft, The Netherlands: ACM, 2016. ISBN: 978-1-4503-4080-9. DOI: 10.1145/2851553.2858676.

[6] M. Frank and M. Hilbrich. "Performance Prediction for Multicore Environments — An Experiment Report". In: *Proceedings of the Symposium on Software Performance (SSP 2016)*. 2016.

[7] M. Frank, M. Hilbrich, S. Lehrig, and S. Becker. "Parallelization, Modeling, and Performance Prediction in the Multi-/Many Core Area: A Systematic Literature Review". In: *Proceedings of the 7th IEEE International Symposium on Cloud and Service Computing (SC2 2017)*. 2017.

[8] M. Frank, F. Klinaku, and S. Becker. "Challenges in Multicore Performance Predictions". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE 2018)*. ACM, 2018.

[9] M. Frank, S. Staude, and M. Hilbrich. "Is the PCM Ready for ACTORs and Multicore CPUs? — A Use Case-based Evaluation". In: *Proceedings of the 8th Symposium on Software Performance (SSP 2017)*. 2017.

[10] C. Heger, A. van Hoorn, M. Mann, and D. Okanović. "Application Performance Management: State of the Art and Challenges for the Future". In: *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2017)*. ACM, 2017. DOI: 10.1145/3030207.3053674.

[11] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger. "Performance Engineering for Microservices: Research Challenges and Directions". In: *Companion of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2017)*. ACM, 2017. ISBN: 978-1-4503-4899-7. DOI: 10.1145/3053600.3053653.

[12] A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Kiel Computer Science Series 2014/6. Dissertation, Faculty of Engineering, Kiel University. Kiel, Germany: Department of Computer Science, Kiel University, 2014. ISBN: 978-3-7357-5118-8.

[13] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[14] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske. "Hora: Architecture-aware online failure prediction". In: *Journal of Systems and Software* (2018). ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.02.041.

[15] H. Schulz, T. Angerstein, and A. van Hoorn. "Towards Automating Representative Load Testing in Continuous Software Engineering". In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE 2018) Companion (7th International Workshop on Load Testing and Benchmarking of Software Systems, LTB 2018)*. ACM, 2018.

[16] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, and H. Krcmar. "WESSBAS: Extraction of Probabilistic Workload Specifications for Load Testing and Performance Prediction—A Model-Driven Approach for Session-Based Application Systems". In: *Journal on Software and System Modeling (SoSyM)* (2018).

[17] J. Walter, A. van Hoorn, H. Koziolek, D. Okanovic, and S. Kounev. "Asking "What?", Automating the "How?": The Vision of Declarative Performance Engineering". In: *Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering (ICPE 2016)*. ACM, 2016.

MetaExp
Investigating techniques for knowledge graphs

Sebastian Bischoff[1], Freya Behrens[2], Julius Rückin[1], and Adrian Ziegler[2]

[1] Technical University of Munich
{first.last}@tum.de
[2] Hasso Plattner Institut
{first.last}@student.hpi.de

In this report, we are describing the development of our knowledge graph exploration prototype MetaExp and the follow up projects investigating specific issues. The follow up projects include meta-path retrieval, active learning, knowledge graph completion and meta-path embedding.

1 Introduction

In the last few years, we have experienced a significant increment in the adoption of *knowledge graphs* to model complex phenomena in various domains. A knowledge graph is a heterogeneous network of entities, such as the actress *Diane Kruger* and the movie *Inglorious Basterds*, connected via named relationships, such as *acted_in*. These networks conveniently represent various information like human knowledge, biological processes, and research work.

Nevertheless, flexibility comes at the cost of complexity, since knowledge graphs usually have no predefined schema, and query languages, such as SPARQL, are hard to understand for novice users. Moreover, commonly available knowledge graphs like YAGO [11] and The Movie Database [20], contain millions of nodes and edges. Hence, analyzing such graphs with traditional tools like Pegasus [8] and SNAP [1] is usually difficult for inexperienced users. As a remedy, exploration tools have been introduced to assist the user formulating queries with simple interfaces [14], or restricting the area of interest to some subset of the nodes [16]. The main shortcomings of such approaches are the inability to adapt to the *individual user preferences* [14] on the one hand, and the lack of a *global view* on the graph [16] on the other hand.

Our research prototype MetaExp [3] combines both. It integrates individual user preferences in specific substructures and concepts that are reflected by abstract-typed paths. Furthermore, it efficiently summarizes global information about concrete instances of prioritized conceptual structures. In detail, our exploration tool offers the following interactions to select user preferences:

- querying sub-patterns or nodes with certain properties and selecting entities of interest for further exploration
- defining importance of abstract-typed paths that connect selected entities and represents conceptual relationships between nodes

Based on these input information, our software derives similarity information between the selected entities as follows:

- expansion of user input about few relevant conceptual relationships to other connecting concepts of similar nature between selected entities
- analysis of occurrences of important and less important conceptual connections and derivation of a similarity between entities based on extracted preferences and covered global structures simultaneously

All in all, our tool enables the user to define prior information about the use case independently of the data instances encoded in the graph and at the same time match this prior with the evidence given by the data for a strong or weak similarity between entities of interest. In the end, the tool derives a similarity score that reflects the degree of similarity between entities with regard to both, prior interests and the global graph data.

2 Project goal

Based on the insights from building and evaluating the research prototype, we derived different areas which need to be improved for fully functional graph exploration tools. This included the following parts

- Efficient computation and retrieval of meta-paths [17, 22]
- User-driven feature learning in knowledge graphs [15]
- Methods for efficient training of neural networks for gathering user feedback in an active learning setting [24]
- Incorporating explanations into active learning [2]
- Evaluation of current knowledge graph completion methods [9]
- A learning procedure for vector representations of meta-paths [5]

We developed and evaluated this tasks experimentally using the 1000 node cluster of the Future SOC Lab and the GPU servers equipped with multiple Tesla K80.

3 Experiments

3.1 User-driven feature learning in knowledge graphs

Heterogeneous knowledge graphs contain rich information in various formats. Due to the natural complexity and size of these knowledge bases researchers try to fit data mining and machine learning techniques to the graph-context in order to perform analysis tasks as link prediction [10]. The success of several machine learning approaches highly depend on the quality of the features describing the graph. A qualitative suitable feature space should reflect similarity or dissimilarity between nodes in order to support the machine learning algorithm to distinguish between e. g. node classes or clusters.

To this end, one can learn features in an unsupervised manner by optimizing an objective function. Inspired by the success of deep learning and explicitly by word embeddings for natural-language processing (NLP) [12], node embedding techniques with the aim to preserve local neighborhood structure optimized by Stochastic Gradient Descent (SGD) appeared [7, 13, 19]. Nevertheless, all methods lack reflecting similarity properties of a node not determined by local neighborhood, but larger sub-structures, roles of nodes or communities [21]. Although node2vec [7] was especially designed to capture global structure, it fails because of difficult hyperparameter tuning bounded to a specific dataset. Thus, these approaches cannot keep pace with the variety of real-world graph structures and different prediction tasks.

VERSE [21] overcomes these problems with the help of a versatile node similarity notion. But in the end, it proposes only similarity measures optimized for capturing graph structure. This type of similarity measure is shown to be not fully suitable for heterogeneous knowledge graphs, since node similarity does not only depend on structure, but is also determined by semantic meanings of different-typed nodes and edges [18]. However, one can integrate a customized similarity measure easily due to the generic framework of VERSE.

With regard to heterogeneous node embeddings, some works like metapath2Vec [6] try to integrate semantics in their node embeddings by limiting graph structure to a particular sequence of node types. Nevertheless, metapath2vec suffers from diverse shortcomings, since it considers only one node type sequence as a heterogeneous semantic and applies the same objective to its learning routine as [7, 13], known to preserve only local structure.

We proposed hete-VERSE, a framework to learn knowledge graph embeddings that incorporates graph structure and detailed expert knowledge about semantic concepts in the graph at the same time. The framework is explicitly built upon the core of VERSE [21]. We share VERSE's perspective that the problem of learning meaningful embeddings breaks down to a suitable node similarity definition. VERSE allows us to reuse the optimization core and to define a customized node similarity distribution, so that the learned embeddings reflect the given similarity measure. Thus, we developed a similarity measure based on meta-path-restricted random walks introduced by metapath2vec [6]. We analyzed softenings and weaknesses regarding the formalization of meta-path-based random walks in the light of our application and adapted it to incorporate user knowledge. Besides, we put our work into context of related work and showed how our approach differs from already existing solutions. Additionally, we developed a reusable experiment framework for meaningful and reproducible node embedding experiments.

Furthermore, we evaluated hete-VERSE on a real-world dataset similar to the dataset used by metapath2vec and compared the performance with other state-of-the-art competitors in different data mining tasks. First, the experiments show that hete-VERSE is highly sensitive to the choice of meta-paths. Second, to some extent hete-VERSE requires also structural similarity information not bounded to a meta-path to be able to learn an embedding as good as or even slightly better, than its competitors. Nevertheless, we examined the signal of different meta-path-preference models raised to the optimization procedure compared to the signal raised by structural similarity. In the end, even in the case of meta-path-based random walks with restart to capture further heterogeneous neighborhoods, we fail in achieving comparably results regarding our competitors. Moreover, we cannot achieve such a significant performance improvement as metapath2vec in a similar setting only by the use of meta-path-based random walks.

3.2 Active learning with curriculum

A successful graph exploration is often dependent on the user's specific preferences and purposes. Therefore, it is inevitable that we think of algorithms that make it easy and effortless for the user to communicate her ideas. Active learning is a paradigm often used to solve this problem. Our specific research question was in how far curriculum learning [4] can be incorporated into active learning and what benefits it brings. The model used for all experiments was a neural network, which we trained with the GPUs provided by the FSOC.

First, we evaluated ways to define a curriculum on meta-paths. The experiments were clearly in favor of the length difficulty measure. The length difficulty measure lets the neural network train on short meta-paths in the beginning. Gradually longer meta-paths are added to the training data, until we train on the whole data set. The intuition behind this procedure is that short meta-paths are easier to learn because they simply have less characteristics. Furthermore, longer meta-paths contain concepts from shorter ones, thereby building up on them. Incorporating shorter meta-paths earlier therefore proved beneficial to learn the label of longer ones. With that strategy, we were able to reduce generalization error significantly in comparison to a no-curriculum strategy.

Having a difficulty measure at hand, we were able to gradually build an active curriculum by prioritizing labelling of shorter meta-paths in the beginning of the active learning process. This resulted in that the query strategy only had to choose from a subset of the unlabelled data, which improved its response time compared to active learning without a curriculum.

Additionally, we examined the learning behavior. We found that there has to be made a trade off between a user's labelling budget and the quality of the model. If the labelling budget is too small, the active curriculum is more harmful than beneficial for lowering the test set error. The more data we can label, the more expressive it becomes. In our experiments, we had to label roughly half of the unlabelled data in order to see the curriculum strategy outperform standard query strategies.

3.3 Incorporating explanations into active learning

We investigated how using explanations from the expert rather than labels could improve the estimated relevance of the paths. In order to be able to retrieve an experts preference, short paths of node and edge types on the graph, so-called meta-paths, are provided by the experts themselves, which present semantic relations on the graph. A model then uses active learning techniques to learn them.

Overall, we provide methods to make expert explanations in form of local linear approximations usable with data explanations. The experiments are conducted on datasets for regression tasks. All of their domains could possibly occur in a setting where experts are available and the labels are costly to obtain. For example, it could be difficult and costly to follow up on the participants of a diabetes study one year later, but when the data is available a doctor could explain the result for one person. In each experiment, the goal is to train a learner model on a regression task, where the performance is measured by the mean squared error on a test dataset or in case of a regression between 0 and 1 with the logarithmic loss. In addition, an expert model is available trained on the complete dataset. For these tasks we always use random forests. Our experimental results are in line with previous work which demonstrates how explanations can improve a model's performance. Beyond the scope of their evaluations, we show that explanations are only beneficial to a model, if the available original instances are less accurate than the provided explanations. In addition, we evaluate on different levels of expert expertise and show that the effectiveness of the explanation also depends on the novelty of the learned concept.

3.4 A learning procedure for vector representations of meta-paths

We studied the problem of feature learning on heterogeneous knowledge graphs. These features can be used to perform tasks such as link prediction, classification and clustering on graphs. Knowledge graphs provide rich semantics encoded in the edge and node types. Meta-paths consist of these types and abstract paths in the graph.

Until now, meta-paths can only be used as categorical features with high redundancy and are therefore unsuitable for machine learning models. We propose meta-path embeddings to solve this problem by learning semantical and compact vector representations of them. Current graph embedding methods only embed nodes and edge types and therefore miss semantics encoded in the combination of them. Our method embeds meta-paths using the skipgram model with an extension to deal with the redundancy and high amount of meta-paths in big knowledge graphs. [5]

We evaluated the developed approach by predicting links on knowledge graphs. Our data source is Wikidata [23] which provides a time-evolving graph with roughly 46 million nodes and a rich schema. We imported the data into neo4j, converted the type annotations into a class hierarchy, searched new edges between two versions, defined embeddings, mined meta-paths and predicted new edges. We evaluated our method with a meta-path-based edge embedding, a meta-path-based node embedding and a node type-based node embedding. The node type-based node embedding works well and is an indicator that the semantics of node types and edge types are captures well. The meta-path-based node embedding either does not get enough information based on the low number of meta-paths per node or does not use this information effectively. The meta-path-based edge embedding suffers from problems computing the meta-paths between the two nodes of an edge.

4 Status & Conclusion

In this project, we developed and evaluated different new ways to support the exploration of large knowledge graphs. Our prototype MetaExp demonstrated different problems with current approaches and showed the most important issues. We approached them in multiple follow up works which are described in this report.

In our next proposed project, we want to develop our approach for meta-path embeddings further by evaluating it more thoroughly and developing solutions for still existing issues.

In the end, we want to thank the Future SOC Lab team especially Bernhard Rabe for supporting us in setting up our experiments.

References

[1] D. A. Bader and K. Madduri. "Snap, small-world network analysis and partitioning: An open-source parallel graph framework for the exploration of large-scale networks". In: *IPDPS*. 2008, pages 1–12.

[2] F. Behrens. "Learning from Expert Knowledge: Enriching Interactive Learning with Explanations". Bachelor's Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[3] F. Behrens, S. Bischoff, P. Ladenburger, J. Rückin, L. Seidel, F. Stolp, M. Vaichenker, A. Ziegler, D. Mottin, F. Aghaei, et al. "MetaExp: Interactive Explanation and Exploration of Large Knowledge Graphs". In: *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee. 2018, pages 199–202.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. ACM, 2009.

[5] S. Bischoff. "Feature Learning for Meta-Paths in Knowledge Graphs". arXiv: 1809.03267. Bachelor's Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[6] Y. Dong, N. V. Chawla, and A. Swami. "metapath2vec: Scalable representation learning for heterogeneous networks". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pages 135–144.

[7] A. Grover and J. Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2016, pages 855–864.

[8] U. Kang, C. E. Tsourakakis, and C. Faloutsos. “Pegasus: A peta-scale graph mining system implementation and observations”. In: *ICDM*. 2009, pages 229–238.

[9] P. Ladenburger. “On Predicting Missing Links in Knowledge Graphs”. Bachelor’s Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[10] L. Lü and T. Zhou. “Link prediction in complex networks: A survey”. In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pages 1150–1170.

[11] F. Mahdisoltani, J. Biega, and F. Suchanek. “Yago3: A knowledge base from multilingual wikipedias”. In: *CIDR*. 2014.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pages 3111–3119.

[13] B. Perozzi, R. Al-Rfou, and S. Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pages 701–710.

[14] R. Pienta, F. Hohman, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. “Visual Graph Query Construction and Refinement”. In: *SIGMOD*. 2017, pages 1587–1590.

[15] J. Rückin. “Similarity Explanation and Exploration in a Heterogeneous Information Network”. Bachelor’s Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[16] S. Seufert, P. Ernst, S. J. Bedathur, S. K. Kondreddi, K. Berberich, and G. Weikum. “Instant Espresso: Interactive Analysis of Relationships in Knowledge Graphs”. In: *WWW Companion*. 2016, pages 251–254.

[17] F. Stolp. “Succinct representation and retrieval of meta-paths from knowledge graphs”. Bachelor’s Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[18] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks”. In: *Proceedings of the VLDB Endowment* 4.11 (2011), pages 992–1003.

[19] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. “Line: Large-scale information network embedding”. In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2015, pages 1067–1077.

[20] TMDb. *The Movie Database*. Accessed: 2018-01-09. 2018. URL: https://www.themoviedb.org/ (last accessed 2018-04-01).

[21] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller. "VERSE: Versatile Graph Embeddings from Similarity Measures". In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2018, pages 539–548.

[22] M. Vaichneker. "Exhaustive and approximated meta-path extraction from heterogeneous information networks". Bachelor's Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[23] D. Vrandečić and M. Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10 (2014), pages 78–85.

[24] A. Ziegler. "Active Learning with Curriculum on Knowledge Graphs". Bachelor's Thesis. Hasso Plattner Institute, University of Potsdam, 2018.

Improving Test Suite Generation by Testing Google Play's Top 1000 Apps

Thomas Vogel[1], Chinh Tran[2], Irene Moser[3], and Lars Grunske[1]

[1] Software Engineering Group
Humboldt-Universität zu Berlin, Germany
thomas.vogel@informatik.hu-berlin.de
[2] Technische Universität Berlin, Germany
[3] Swinburne University of Technology, Australia

Automated generation of test suites tackles the complexity of testing apps. The selection of the underlying search algorithm is typically done in a trial-and-error fashion due to unknown characteristics of the search problem. Analyzing the fitness landscapes when testing Google Play's top 1000 apps, we want to understand the search problem and improve the search algorithm for generating test suites. In this report, we discuss the research problem and initial steps we have taken in the Future SOC Lab.

1 Introduction

Given the complexity of software, a trend is to automate software engineering tasks using search-based optimization methods, which is known as search-based software engineering [4]. Software testing is a crucial part of software engineering that provides evidence for the correctness of software systems. Due to the costs and time-consuming nature of testing, search-based methods have been successfully used in solving software testing problems [3, 7] and finding faults in existing software [2, 6]. For instance, the search-based test suite generation tool *Sapienz* [6] discovered bugs (in terms of crashes) in roughly every third Android application (app) that has been tested. This shows the practical relevance of testing and the need to continue research in improving testing approaches such as test suite generation.

2 Research Problem

A possibility to improve test suite generation is to address the problem of selecting and configuring the search algorithm for finding optimal test suites. Typically, selection and configuration are done in a black-box fashion since there is missing knowledge about the characteristics of the search problem and thus about the expected performance of an algorithm. Thus, either different algorithms are tested in a trial-and-error approach to see which algorithm and configuration are most suitable for the specific problem, or a popular algorithm with its default configuration is just selected. While the former is costly and time-consuming, the latter does not guarantee that the selected algorithm will perform well due to the *No-Free-Lunch theorems* *for search and optimization* [9, 10]. These theorems proof that there is no search/optimization algorithm that is at all times superior to another algorithm. Thus, for each specific problem, an algorithm must be selected and configured.

3 Goal

The goal of our project is to better understand the search problem such that we can make an informed decision for selecting and configuring the search algorithm, which should eventually yield better results. To obtain a better understanding of a search problem, we use *fitness landscape analysis* [5, 8] techniques that investigate the search space and the interaction of an algorithm with the search space. Feedback from the analysis will guide the selection and configuration of the algorithm.

In this project, we built upon our previous work [1] and focus on the automated test suite generation and testing of Android apps. We plan a large study to investigate the test suite generation problem for the top 1000 apps of the Google Play store and to improve the state-of-the-art approach Sapienz [6]. Sapienz uses a *default* genetic algorithm with its *default* configuration to automatically generate test suites. Hence, Sapienz does not exploit any knowledge about the search problem that could be obtained by fitness landscape analyses for the apps under test.

In addition to aiming at improving the state of the art, this project will eventually test the top 1000 apps and therefore provide as well a report about the state of the practice in how reliable or faulty today's most popular Android apps are.

4 Findings at the Future SOC Lab

In the following, we discuss the used Future SOC Lab resources and the deployment and distribution model we have identified for our upcoming studies in the lab.

4.1 Resources

Initially, we aimed for an environment with virtual machines (VMs), which, however, turned out to be inefficient. To test an app, it is required to run the app inside an emulator of a mobile device hosting the Android operating system. Such an emulator is itself a VM. Running such an emulator inside of a VM of the Future SOC Lab, the emulator cannot benefit from any VM/hardware acceleration. To illustrate the inefficacy of the virtualized environment, booting an emulator in the Future SOC Lab was ten times slower than on a standard notebook. Similarly, the execution of the generated test suites on an app running in an emulator would be slowed down drastically, which prevents testing at a scale of 1000 apps.

As a consequence, the Future SOC Lab has provided us native access to eight blade servers. We used these severs for initial experiments to identify a suitable deployment and distribution model for the large-scale study of testing 1000 apps.

4.2 Deployment and Distribution Model

The resource-intensive part of generating test suites with search-based methods is the fitness evaluation of test suites. Calculating the fitness of a test suite consists of identifying (1) whether the test suite reveals a fault in the app under test, (2) the coverage achieved by the test suite, and (3) the length of the test sequences contained in the suite (cf. [6]). To obtain (1) and (2), a test suite must be executed to the app under test that is deployed and running in an Android emulator.

Using a population-based search algorithm, particularly a genetic algorithm (GA) as Sapienz does, multiple test suites are *varied* and must be *evaluated* in terms of calculating their fitness at the same time to *select* the best test suites for each generation of the evolutionary search. We are currently considering a population of 50 test suites so that up to 50 test suites are evaluated at the same time. At this point, we parallelize the fitness evaluation by booting multiple emulators (currently, up to ten) on one server and deploying the app under test to all of them. Thus, ten test suites can be executed concurrently to evaluate their fitness (see Figure 1).

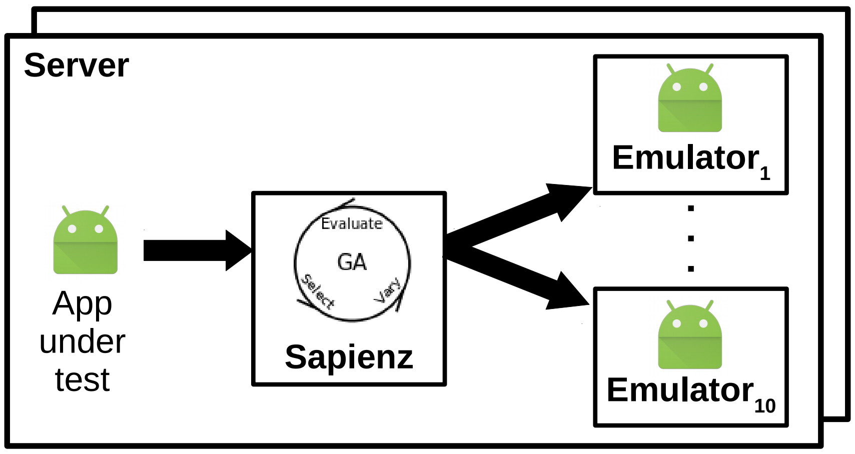


Figure 1: Deployment and Distribution Model

Since the test suite generation process of Sapienz itself cannot be distributed, each server runs one instance of Sapienz to test an individual app while each instance exploits parallelization by using multiple emulators. Thus, we use the servers to run Sapienz concurrently for different apps and the resources of one server to parallelize the execution of test suites for an individual app. We hope that this deployment and distribution model allows us to scale our study to 1000 apps.

Our initial experiments show that the test suite generation process with a population size of 50 test suites, each containing 5 test sequences with 20–500 events, and over 50 iterations (generations) takes around 4–6 hours for one app. Considering these costs, we will likely constrain the test suite generation to a time budget, for instance, a budget of 30 minutes has been used by Mao et al. [6].

5 Conclusion and Next Steps

In this report, we have discussed the research problem of selecting and configuring search algorithms, specific to each app under test, based on fitness landscape analysis to improve the generation of test suites. The goal is to obtain better test suites with respect to fault revelation, coverage, and length of test sequences.

As the next steps, we will run the large-scale empirical study at the Future SOC Lab that compares head-to-head the original Sapienz with our extended Sapienz that includes techniques for fitness landscape analysis and for statically and dynamically adjusting the search algorithm. These adjustments adapt the search strategy for finding optimal test suites. The study will be conducted in two steps, first using a 68 app benchmark before testing Google Play's top 1000 apps.

References

[1] A. Aleti, I. Moser, and L. Grunske. "Analysing the fitness landscape of search-based software testing problems". In: *Autom. Softw. Eng.* 24.3 (2017), pages 603–621. DOI: 10.1007/s10515-016-0197-7.

[2] G. Fraser and A. Arcuri. "1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite". In: *Empirical Software Engineering* 20.3 (2015), pages 611–639. DOI: 10.1007/s10664-013-9288-2.

[3] M. Harman. "The Current State and Future of Search Based Software Engineering". In: *Workshop on the Future of Software Engineering, FOSE*. 2007, pages 342–357. DOI: 10.1109/FOSE.2007.29.

[4] M. Harman, S. A. Mansouri, and Y. Zhang. "Search-based Software Engineering: Trends, Techniques and Applications". In: *ACM Comput. Surv.* 45.1 (2012), 11:1–11:61. DOI: 10.1145/2379776.2379787.

[5] K. M. Malan and A. P. Engelbrecht. "A survey of techniques for characterising fitness landscapes and some possible ways forward". In: *Information Sciences* 241.Suppl. C (2013), pages 148–163. DOI: 10.1016/j.ins.2013.04.015.

[6] K. Mao, M. Harman, and Y. Jia. "Sapienz: Multi-objective Automated Testing for Android Applications". In: *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ISSTA 2016. ACM, 2016, pages 94–105. DOI: 10.1145/2931037.2931054.

[7] P. McMinn. "Search-based Software Test Data Generation: A Survey". In: *Software Testing, Verification and Reliability* 14.2 (2004), pages 105–156. DOI: 10.1002/stvr.v14:2.

[8] E. Pitzer and M. Affenzeller. "A Comprehensive Survey on Fitness Landscape Analysis". In: *Recent Advances in Intelligent Engineering Systems*. Springer, 2012, pages 161–191. DOI: 10.1007/978-3-642-23229-9_8.

[9] D. H. Wolpert and W. G. Macready. “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pages 67–82. DOI: 10.1109/4235.585893.

[10] D. H. Wolpert and W. G. Macready. *No Free Lunch Theorems for Search*. Technical report SFI-TR-95-02-010. Santa Fe Institute, 1995.

High-Performance Property Graph Queries

Gábor Szárnyas[1], József Marton[2], János Maginecz, and Dániel Varró[1]

[1] MTA-BME Lendület Cyber-Physical Systems Research Group
Department of Measurement and Information Systems
Budapest University of Technology and Economics
szarnyas@mit.bme.hu, varro@mit.bme.hu
[2] Database Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
marton@db.bme.hu

The property graph data model of modern graph database systems is increasingly adapted for storing and processing heterogeneous datasets like networks. Many challenging applications with near real-time requirements — e. g. financial fraud detection, recommendation systems, and on-the-fly validation — can be captured with graph queries, which are evaluated repeatedly. To ensure quick response time for a changing data set, these applications would benefit from applying incremental view maintenance (IVM) techniques, which can perform continuous evaluation of queries and calculate the changes in the result set upon updates.

While IVM problems have been studied extensively over relational databases, views on property graph queries require operators outside the scope of standard relational algebra. As such, currently, no industrial graph databases provide support for incremental views and the feature coverage of IVM research tools is limited to core operators. In this report, we present an overview of the challenges related with IVM on property graph queries, survey related work and highlight a possible approach.

1 Problem Statement

Graph processing problems are common in modern database systems, where the *property graph* (PG) data model [2, 3, 19, 30, 39] is gaining widespread adoption. Property graphs extend labelled graphs with properties for both vertices and edges. Compared to previous graph modelling approaches, such as the RDF data model (which represents properties as triples), PGs allow users to store their graphs in a more compact and comprehensible representation. Due to the novelty of the PG data model, no standard query language has emerged yet. The industry-driven *openCypher initiative* aims to standardize the Cypher language [19] of the Neo4j graph database. The openCypher language uses a SQL-like syntax and combines graph pattern matching with relational operators (aggregations, filtering, projection, etc.). In this work, we target queries specified in the openCypher language.

2 Motivation

Numerous use cases of graph databases rely on complex queries and require low response time for repeated executions, including financial fraud detection, and recommendation engines. In addition, graph databases are increasingly used in software engineering context as a semantic knowledge base for model validation [6, 16, 51], source code analysis [29], etc. Users of these scenarios could greatly benefit from an incremental query engine that allows them to *register views* on property graphs and *maintain their state* continuously upon changes.

A recent survey on graph processing [48] also concluded that there is industrial need for incremental (and streaming) graph query engines. This investigation also revealed that currently no such systems exist, with the exception of Graphflow (an active graph database prototype [33]) which shares many authors with the survey. Our own observation, which is based upon using these tools in an open property graph and RDF benchmark [53], is that incremental evaluation is only supported for very basic language features (practically, only 4-5 queries out of a total 25 queries). This clearly indicates that *incremental view maintenance is still a major research challenge in the context of property graphs*.

3 Findings

In relational database systems, *incremental view maintenance* (IVM) techniques have been used for decades for repeated evaluation of a predefined query set on continuously changing data [9, 18, 23, 24, 25, 27, 28, 35, 40, 50, 54]. However, these techniques typically build on assumptions that do not hold for PG queries. Incremental PG queries present numerous challenges:

1. *Schema-optional data model.* Existing IVM techniques assume that the database schema is a priori known. While this is a realistic assumption for relational databases, the data model of most property graph systems is schema-free or schema-optional at best [11, 19]. Hence, to use IVM, users are required to manually define the schema of the graph, which is tedious and error-prone.
2. *Nested data structures.* Most IVM techniques assume relational data model with 1NF relations. However, the property graph data model defines rich structures, including the properties on graph elements and paths. Collection types, such as sets, bags, lists, and maps are also allowed [3, 19]. These can be represented as NF^2 (non-first normal form) data structures, but their mapping to 1NF relations is a complex challenge.
3. *Mix of instance- and meta-level data.* Queries access not only data fields from the instance graph, but also metadata such as vertex/edge labels [19, 47].
4. *Handling null values and outer joins.* PG queries allow null values and optional pattern matches, similarly to outer joins in relational databases. Most IVM works do not consider this challenge, except [20], [26], and [36].

5. *Complex aggregations.* PG queries allow complex aggregations, e.g. aggregations on aggregations [41] and using non-distributive aggregation functions (e.g. min, max, stdev) which are difficult to calculate incrementally [44].

6. *Mix of queries and transformations.* Some PG query languages (e.g. openCypher) allow combining update operations with queries. Most traditional IVM techniques do not consider this challenge, and omit related issues such as conflict set resolution. *Discrimination networks* from rule-based expert systems are better suited to handle this issue [18, 28, 40].

7. *List handling.* Property graph data sets and queries make use of lists both as a way to store collection of primitive values and to represent paths in the graphs. Order-preserving techniques have only been studied in the context of IVM on XQuery expressions [17], for trees but not for graphs.

8. *Reachability queries.* Unbounded reachability queries on graphs with few connected components need to calculate large transitive closures, which makes them inherently expensive [7]. Hence, the impact of the IVM on reachability is more limited compared to non-recursive queries and using space-time tradeoff techniques is more expensive.

9. *Skewed data distribution.* Subgraph matching is often implemented as a series of binary joins. Recent work [42] revealed that binary joins are inefficient on data sets with skewed distributions of certain edge types (displayed by graph instances in many fields, e.g. in social networks). Hence, a large body of new research proposes multiway joins to achieve theoretically optimal complexity.

10. *Higher-order queries.* PG queries often employ *higher-order* expressions [12], e.g. processing the vertices/edges on a path (also known as *path unwinding* [4]). Incrementalization of higher-order expressions is not yet studied in depth [13], and currently there are no implementations using such techniques.

In our work, we so far addressed challenges 1–5 while leaving 6–10 for future work for the database community. Note that research tools with IVM support typically cover only 1 and 2 while industrial PG query tools do not support IVM.

4 Contributions

In our detailed technical report [52], we show how to reduce the challenge of incremental graph evaluation of openCypher queries over property graphs to traditional relational algebra to enable the use of existing IVM techniques. In particular:

- We introduce extensions for relational algebra (RA) in order to handle graph-specific operators. We use the resulting *graph RA* (GRA) to capture the semantics of a large subset of the openCypher language.

- We define a mapping for PG data to nested relations, and transform the queries to *nested RA* (NRA). The data model can represent both the property graph and the resulting tables, while the NRA operators have sufficient expressive power to capture operations on the PG. This allows the algebra to be *composable and closed* [49].
- We define a transformation chain to translate the nested algebraic query plans to *flat RA* (FRA) expressions. To this end, we present a *schema inferencing algorithm* that eliminates the need to define the graph schema in advance.
- We propose an architecture for IVM for PG queries based on the Rete algorithm [18]. As a proof of concept prototype, we present the *ingraph* tool which is capable of evaluating openCypher graph queries incrementally.[3]

5 Related Work

Covering the rich set of features required by property graph queries—ranging from expressing negative structural conditions to unnesting and reachability queries—requires multiple IVM algorithms. Surveys on IVM approaches were presented in paper [23], book [24], and monograph [14]. However, even such comprehensive surveys did not cover challenges 1–3 and 8–10 raised in section 3.

One of the first takes on algebraic view maintenance was presented in [45]. Its algorithm was improved by Griffin and Libkin [22], who also produced one of the seminal papers in the field [21]. Later studies added extensions for additional operators: aggregations [46], semijoins/outer joins [20], order-preserving maintenance [17], and outer joins/aggregations [36]. Techniques for IVM on object-oriented data were presented in [38]. The authors of [31] described the *Dynamic Yannakakis Algorithm* (Dyn), an incremental join algorithm targeting acyclic queries similarly to the well-known *Yannakakis Algorithm*. A superset of the authors later extended Dyn to a generalized notion of acyclicity in [32].

Due to the rise of interest in efficient graph processing techniques, many recent works targeted the evaluation of cyclic queries. One such example is the *triangle query*, which looks for three-cliques in the graph. It can be shown [42] that it is impossible to optimally evaluate this query using only binary joins. In fact, such solutions are asymptotically suboptimal. Hence, executing this query on large graph-like data sets is a challenge for most contemporary database systems (including RDBMSs and graph databases). The family of *worst-case optimal join algorithms* [42] aims to solve such queries efficiently. Such algorithms include the *Leapfrog Triejoin algorithm* of the (Datalog-based) LogicBlox system [5], whose incremental variant was described in [56]. An incrementalized worst-case optimal join algorithm named *delta generic join* with strong theoretical guarantees was recently presented in [1]. Table 1 shows

[3]ingraph is available as an open-source tool at http://github.com/ftsrg/ingraph.

an overview of IVM techniques and their applicability to bags, NF^2 data, null values, complex aggregations, ordering, and support of subgraph matching-style queries.

Table 1: Overview of related literature on IVM techniques. *NF^2*: non-first normal form relations, *agg.*: aggregation, *ord.*: ordering, *gra.* = subgraph matching queries; ⊗ fully supported, ∅ supported to some extent, ○ not supported.

ref.	contributions	bag	NF^2	null	agg.	ord.	gra.
[9]	determining irrelevant updates, maintenance of select–project–join views	○	○	○	○	○	○
[45]	change propagation equations for relational algebra	○	○	○	○	○	○
[25]	counting algorithm (non-recursive views), DRed algorithm (recursive views)	⊗	○	○	○	○	○
[15]	change propagation equations for bag alg., incl. aggregation but no group-by	⊗	○	○	∅	○	○
[34]	extending IVM techniques to maintain views defined over a nested data model	⊗	⊗	○	○	○	○
[37]	maintaining the transitive closure of directed graphs using a SQL-like language	⊗	⊗	○	⊗	○	○
[41]	group-by-aggregation, summary-deltas for representing changes	⊗	○	○	⊗	○	○
[22]	improved change propagation equations for RA, cf. [22]	○	○	○	○	○	○
[20]	change propagation equations for semijoins, antijoins and outer joins	○	○	⊗	○	○	○
[44]	maintenance of non-distributive aggregate functions	⊗	○	⊗	⊗	○	○
[38]	incremental equations for the operators of the nested model	○	⊗	○	○	○	○
[17]	order-preserving maintenance of XQuery views	⊗	⊗	○	∅	⊗	○
[58]	IVM on top-*k* views	○	○	○	○	∅	○
[26]	generalized summary-deltas, group-by-aggregations, outer joins	⊗	○	⊗	⊗	○	○
[36]	outer joins and aggregation, fixes [26]	⊗	○	⊗	⊗	○	○
[56]	incremental maintenance for Leapfrog Triejoin	○	○	○	○	○	⊗
[35]	higher-order IVM, viewlet transformations, the DBToaster system	⊗	○	∅	∅	∅	○
[31]	Dynamic Yannakakis Algorithm for incremental evaluation of acyclic queries	⊗	○	○	⊗	○	○
[43]	factorized IVM	⊗	○	○	○	○	○
[32]	generalized Dynamic Yannakakis algorithm	⊗	○	○	○	○	∅
[1]	delta-generic join for subgraph matching, particularly cyclic queries	○	○	○	○	○	⊗

5.1 Rule-based Expert Systems

IVM has been used extensively in the context of *rule-based expert systems* (also known as *production systems*), and were supported by *discrimination networks*. Notable approaches include Rete [18], TREAT [40], and Gator [28] (**g**ener**a**lized **T**REAT **o**r **R**ete). In expert systems, users formulate *rules* (or *productions*), which have a *left-hand side* (LHS) and a *right-hand side* (RHS). As described in [40], a *rule engine* (or *production system interpreter*) repeatedly executes a cycle of three operations: (1) *match* working memory elements based on the LHS, (2) *resolve* the conflict set, and (3) *act* based on the RHS. The paper notes that the first step, *match*, is considered the bottleneck in most production systems, argues for using IVM techniques, and presents the TREAT algorithm. A performance comparison of the Rete and TREAT algorithms was later given in [57] and [10], concluding that TREAT is favourable in many cases, but noting that it is more difficult to implement.

The Viatra Query system performs IVM on graph models [55]. Its Viatra Query Language (VQL) is based on Datalog [8], and supports recursive queries, subpattern calls, along with some aggregations. Viatra uses the Rete algorithm to perform efficient model validation and transformation operations over graphs [54].

References

[1] K. Ammar, F. McSherry, S. Salihoglu, and M. Joglekar. "Distributed Evaluation of Subgraph Queries Using Worst-case Optimal and Low-Memory Dataflows". In: *Proc. VLDB Endow.* 11.6 (2018), pages 691–704. DOI: 10.14778/3184470.3184473.

[2] R. Angles. "The Property Graph Database Model". In: *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*. Edited by D. Olteanu and B. Poblete. Volume 2100. CEUR Workshop Proceedings. CEUR-WS.org, 2018.

[3] R. Angles, M. Arenas, P. Barceló, P. A. Boncz, G. H. L. Fletcher, C. Gutiérrez, T. Lindaaker, M. Paradies, S. Plantikow, J. F. Sequeda, O. van Rest, and H. Voigt. "G-CORE: A Core for Future Graph Query Languages". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. Edited by G. Das, C. M. Jermaine, and P. A. Bernstein. ACM, 2018, pages 1421–1432. DOI: 10.1145/3183713.3190654.

[4] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc. "Foundations of Modern Query Languages for Graph Databases". In: *ACM Comput. Surv.* 50.5 (2017), 68:1–68:40. DOI: 10.1145/3104031.

[5] M. Aref, B. ten Cate, T. J. Green, B. Kimelfeld, D. Olteanu, E. Pasalic, T. L. Veldhuizen, and G. Washburn. "Design and Implementation of the LogicBlox System". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. Edited by T. K. Sellis, S. B. Davidson, and Z. G. Ives. ACM, 2015, pages 1371–1382. DOI: 10.1145/2723372.2742796.

[6] G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös. "Incremental Evaluation of Model Queries over EMF Models". In: *Model Driven Engineering Languages and Systems - 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Proceedings, Part I*. Edited by D. C. Petriu, N. Rouquette, and Ø. Haugen. Volume 6394. Lecture Notes in Computer Science. Springer, 2010, pages 76–90. DOI: 10.1007/978-3-642-16145-2_6.

[7] G. Bergmann, I. Ráth, T. Szabó, P. Torrini, and D. Varró. "Incremental Pattern Matching for the Efficient Computation of Transitive Closure". In: *Graph Transformations - 6th International Conference, ICGT 2012, Bremen, Germany, September 24-29, 2012. Proceedings*. Edited by H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg. Volume 7562. Lecture Notes in Computer Science. Springer, 2012, pages 386–400. DOI: 10.1007/978-3-642-33654-6_26.

[8] G. Bergmann, Z. Ujhelyi, I. Ráth, and D. Varró. "A Graph Query Language for EMF Models". In: *Theory and Practice of Model Transformations - 4th International Conference, ICMT@TOOLS 2011, Zurich, Switzerland, June 27-28, 2011. Proceedings*. Edited by J. Cabot and E. Visser. Volume 6707. Lecture Notes in

Computer Science. Springer, 2011, pages 167–182. DOI: 10.1007/978-3-642-21732-6_12.

[9] J. A. Blakeley, P.-Å. Larson, and F. W. Tompa. “Efficiently Updating Materialized Views”. In: *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 28-30, 1986*. Edited by C. Zaniolo. ACM Press, 1986, pages 61–71. DOI: 10.1145/16894.16861.

[10] D. A. Brant, T. Grose, B. J. Lofaso, and D. P. Miranker. “Effects of Database Size on Rule System Performance: Five Case Studies”. In: *17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings*. Edited by G. M. Lohman, A. Sernadas, and R. Camps. Morgan Kaufmann, 1991, pages 287–296.

[11] K. Braunschweig, M. Thiele, and W. Lehner. “A Flexible Graph-Based Data Model Supporting Incremental Schema Design and Evolution”. In: *Current Trends in Web Engineering - Workshops, Doctoral Symposium, and Tutorials, Held at ICWE 2011, Paphos, Cyprus, June 20-21, 2011. Revised Selected Papers*. Edited by A. Harth and N. Koch. Volume 7059. Lecture Notes in Computer Science. Springer, 2011, pages 302–306. DOI: 10.1007/978-3-642-27997-3_29.

[12] P. Buneman, S. A. Naqvi, V. Tannen, and L. Wong. “Principles of Programming with Complex Objects and Collection Types”. In: *Theor. Comput. Sci.* 149.1 (1995), pages 3–48. DOI: 10.1016/0304-3975(95)00024-Q.

[13] Y. Cai, P. G. Giarrusso, T. Rendel, and K. Ostermann. “A theory of changes for higher-order languages: incrementalizing λ-calculi by static differentiation”. In: *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*. Edited by M. F. P. O'Boyle and K. Pingali. ACM, 2014, pages 145–155. DOI: 10.1145/2594291.2594304.

[14] R. Chirkova and J. Yang. “Materialized Views”. In: *Found. Trends Databases* 4.4 (2012), pages 295–405. DOI: 10.1561/1900000020.

[15] L. S. Colby, T. Griffin, L. Libkin, I. S. Mumick, and H. Trickey. “Algorithms for Deferred View Maintenance”. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*. Edited by H. V. Jagadish and I. S. Mumick. ACM Press, 1996, pages 469–480. DOI: 10.1145/233269.233364.

[16] G. Daniel, G. Sunyé, A. Benelallam, M. Tisi, Y. Vernageau, A. Gómez, and J. Cabot. “NeoEMF: A multi-database model persistence framework for very large models”. In: *Sci. Comput. Program.* 149 (2017), pages 9–14. DOI: 10.1016/j.scico.2017.08.002.

[17] K. Dimitrova, M. El-Sayed, and E. A. Rundensteiner. “Order-Sensitive View Maintenance of Materialized XQuery Views”. In: *Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings*. Edited by I.-Y. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann. Volume 2813. Lecture Notes in Computer Science. Springer, 2003, pages 144–157. DOI: 10.1007/978-3-540-39648-2_14.

[18] C. Forgy. “Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem”. In: *Artif. Intell.* 19.1 (1982), pages 17–37. doi: 10.1016/0004-3702(82)90020-0.

[19] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. “Cypher: An Evolving Query Language for Property Graphs”. In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. Edited by G. Das, C. M. Jermaine, and P. A. Bernstein. ACM, 2018, pages 1433–1445. doi: 10.1145/3183713.3190657.

[20] T. Griffin and B. Kumar. “Algebraic Change Propagation for Semijoin and Outerjoin Queries”. In: *SIGMOD Rec.* 27.3 (1998), pages 22–27. doi: 10.1145/290593.290597.

[21] T. Griffin and L. Libkin. “Incremental Maintenance of Views with Duplicates”. In: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995*. Edited by M. J. Carey and D. A. Schneider. ACM Press, 1995, pages 328–339. doi: 10.1145/223784.223849.

[22] T. Griffin, L. Libkin, and H. Trickey. “An Improved Algorithm for the Incremental Recomputation of Active Relational Expressions”. In: *IEEE Trans. Knowl. Data Eng.* 9.3 (1997), pages 508–511. doi: 10.1109/69.599937.

[23] A. Gupta and I. S. Mumick. “Maintenance of Materialized Views: Problems, Techniques, and Applications”. In: *IEEE Data Eng. Bull.* 18.2 (1995), pages 3–18.

[24] A. Gupta and I. S. Mumick. “Maintenance of Materialized Views: Problems, Techniques, and Applications”. In: *Materialized Views: Techniques, Implementations, and Applications*. Cambridge, MA, USA: MIT Press, 1999, pages 145–157. isbn: 0-262-57122-6.

[25] A. Gupta, I. S. Mumick, and V. S. Subrahmanian. “Maintaining Views Incrementally”. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*. Edited by P. Buneman and S. Jajodia. ACM Press, 1993, pages 157–166. doi: 10.1145/170035.170066.

[26] H. Gupta and I. S. Mumick. “Incremental maintenance of aggregate and outerjoin expressions”. In: *Inf. Syst.* 31.6 (2006), pages 435–464. doi: 10.1016/j.is.2004.11.011.

[27] E. N. Hanson. “The Design and Implementation of the Ariel Active Database Rule System”. In: *IEEE Trans. Knowl. Data Eng.* 8.1 (1996), pages 157–172. doi: 10.1109/69.485644.

[28] E. N. Hanson, S. Bodagala, and U. Chadaga. “Trigger Condition Testing and View Maintenance Using Optimized Discrimination Networks”. In: *IEEE Trans. Knowl. Data Eng.* 14.2 (2002), pages 261–280. doi: 10.1109/69.991716.

[29] N. Hawes, B. Barham, and C. Cifuentes. “Frappé: Querying the Linux Kernel Dependency Graph”. In: *Proceedings of the Third International Workshop on Graph Data Management Experiences and Systems, GRADES 2015, Melbourne, VIC, Australia, May 31 - June 4, 2015*. Edited by J. L. Larriba-Pey and T. L. Willke. ACM, 2015, 4:1–4:6. DOI: 10.1145/2764947.2764951.

[30] J. Hölsch and M. Grossniklaus. “An Algebra and Equivalences to Transform Graph Patterns in Neo4j”. In: *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016*. Edited by T. Palpanas and K. Stefanidis. Volume 1558. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[31] M. Idris, M. Ugarte, and S. Vansummeren. “The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates”. In: *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. Edited by S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu. ACM, 2017, pages 1259–1274. DOI: 10.1145/3035918.3064027.

[32] M. Idris, M. Ugarte, S. Vansummeren, H. Voigt, and W. Lehner. “Conjunctive Queries with Inequalities Under Updates”. In: *Proc. VLDB Endow.* 11.7 (2018), pages 733–745. DOI: 10.14778/3192965.3192966.

[33] C. Kankanamge, S. Sahu, A. Mhedhbi, J. Chen, and S. Salihoglu. “Graphflow: An Active Graph Database”. In: *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. Edited by S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu. ACM, 2017, pages 1695–1698. DOI: 10.1145/3035918.3056445.

[34] A. Kawaguchi, D. F. Lieuwen, I. S. Mumick, and K. A. Ross. “Implementing Incremental View Maintenance in Nested Data Models”. In: *Database Programming Languages, 6th International Workshop, DBPL-6, Estes Park, Colorado, USA, August 18-20, 1997, Proceedings*. Edited by S. Cluet and R. Hull. Volume 1369. Lecture Notes in Computer Science. Springer, 1997, pages 202–221. DOI: 10.1007/3-540-64823-2_12.

[35] C. Koch, Y. Ahmad, O. Kennedy, M. Nikolic, A. Nötzli, D. Lupei, and A. Shaikhha. “DBToaster: higher-order delta processing for dynamic, frequently fresh views”. In: *VLDB J.* 23.2 (2014), pages 253–278. DOI: 10.1007/s00778-013-0348-4.

[36] P.-Å. Larson and J. Zhou. “Efficient Maintenance of Materialized Outer-Join Views”. In: *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*. Edited by R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis. IEEE Computer Society, 2007, pages 56–65. DOI: 10.1109/ICDE.2007.367851.

[37] L. Libkin and L. Wong. “Incremental Recomputation of Recursive Queries with Nested Sets and Aggregate Functions”. In: *Database Programming Languages, 6th International Workshop, DBPL-6, Estes Park, Colorado, USA, August 18-20, 1997, Proceedings*. Edited by S. Cluet and R. Hull. Volume 1369. Lecture

Notes in Computer Science. Springer, 1997, pages 222–238. DOI: 10.1007/3-540-64823-2_13.

[38] J. Liu, M. W. Vincent, and M. K. Mohania. "Maintaining Views in Object-Relational Databases". In: *Knowl. Inf. Syst.* 5.1 (2003), pages 50–82. DOI: 10.1007/s10115-002-0067-z.

[39] J. Marton, G. Szárnyas, and D. Varró. "Formalising openCypher Graph Queries in Relational Algebra". In: *Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings*. Edited by M. Kirikova, K. Nørvåg, and G. A. Papadopoulos. Volume 10509. Lecture Notes in Computer Science. Springer, 2017, pages 182–196. DOI: 10.1007/978-3-319-66917-5_13.

[40] D. P. Miranker and B. J. Lofaso. "The Organization and Performance of a TREAT-Based Production System Compiler". In: *IEEE Trans. Knowl. Data Eng.* 3.1 (1991), pages 3–10. DOI: 10.1109/69.75882.

[41] I. S. Mumick, D. Quass, and B. S. Mumick. "Maintenance of Data Cubes and Summary Tables in a Warehouse". In: *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. Edited by J. Peckham. ACM Press, 1997, pages 100–111. DOI: 10.1145/253260.253277.

[42] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. "Worst-case Optimal Join Algorithms". In: *J. ACM* 65.3 (2018), 16:1–16:40. DOI: 10.1145/3180143.

[43] M. Nikolic and D. Olteanu. "Incremental View Maintenance with Triple Lock Factorization Benefits". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. Edited by G. Das, C. M. Jermaine, and P. A. Bernstein. ACM, 2018, pages 365–380. DOI: 10.1145/3183713.3183758.

[44] T. Palpanas, R. Sidle, R. Cochrane, and H. Pirahesh. "Incremental Maintenance for Non-Distributive Aggregate Functions". In: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*. Morgan Kaufmann, 2002, pages 802–813. DOI: 10.1016/B978-155860869-6/50076-7.

[45] X. Qian and G. Wiederhold. "Incremental Recomputation of Active Relational Expressions". In: *IEEE Trans. Knowl. Data Eng.* 3.3 (1991), pages 337–341. DOI: 10.1109/69.91063.

[46] D. Quass. "Maintenance Expressions for Views with Aggregation". In: *Workshop on Materialized Views: Techniques and Applications, VIEWS@SIGMOD 1996. Le Centre Sheraton Hotel, Monteal, Canada, Friday, June 7, 1996*. 1996, pages 110–118.

[47] O. van Rest, S. Hong, J. Kim, X. Meng, and H. Chafi. "PGQL: a property graph query language". In: *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*. Edited by P. A. Boncz and J. L. Larriba-Pey. ACM, 2016, page 7. DOI: 10.1145/2960414.2960421.

[48] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu. "The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing". In: *Proc. VLDB Endow.* 11.4 (2017), pages 420–431. DOI: 10.1145/3186728.3164139.

[49] G. Szárnyas. "Incremental View Maintenance for Property Graph Queries". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. Edited by G. Das, C. M. Jermaine, and P. A. Bernstein. ACM, 2018, pages 1843–1845. DOI: 10.1145/3183713.3183724.

[50] G. Szárnyas, B. Izsó, I. Ráth, D. Harmath, G. Bergmann, and D. Varró. "IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud". In: *Model-Driven Engineering Languages and Systems - 17th International Conference, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014. Proceedings*. Edited by J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfrán. Volume 8767. Lecture Notes in Computer Science. Springer, 2014, pages 653–669. DOI: 10.1007/978-3-319-11653-2_40.

[51] G. Szárnyas, B. Izsó, I. Ráth, and D. Varró. "The Train Benchmark: cross-technology performance evaluation of continuous model queries". In: *Softw. Syst. Model.* 17.4 (2018), pages 1365–1393. DOI: 10.1007/s10270-016-0571-8.

[52] G. Szárnyas, J. Marton, J. Maginecz, and D. Varró. "Reducing Property Graph Queries to Relational Algebra for Incremental View Maintenance". In: *CoRR* abs/1806.07344 (2018). arXiv: 1806.07344.

[53] G. Szárnyas, A. Prat-Pérez, A. Averbuch, J. Marton, M. Paradies, M. Kaufmann, O. Erling, P. A. Boncz, V. Haprian, and J. B. Antal. "An early look at the LDBC social network benchmark's business intelligence workload". In: *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Houston, TX, USA, June 10, 2018*. Edited by A. Arora, A. Bhattacharya, G. H. L. Fletcher, J. L. Larriba-Pey, S. Roy, and R. West. ACM, 2018, 9:1–9:11. DOI: 10.1145/3210259.3210268.

[54] Z. Ujhelyi, G. Bergmann, Á. Hegedüs, Á. Horváth, B. Izsó, I. Ráth, Z. Szatmári, and D. Varró. "EMF-IncQuery: An integrated development environment for live model queries". In: *Sci. Comput. Program.* 98 (2015), pages 80–99. DOI: 10.1016/j.scico.2014.01.004.

[55] D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, and Z. Ujhelyi. "Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework". In: *Softw. Syst. Model.* 15.3 (2016), pages 609–629. DOI: 10.1007/s10270-016-0530-4.

[56] T. L. Veldhuizen. “Incremental Maintenance for Leapfrog Triejoin”. In: *CoRR* abs/1303.5313 (2013). arXiv: 1303.5313.

[57] Y.-W. Wang and E. N. Hanson. “A Performance Comparison of the Rete and TREAT Algorithms for Testing Database Rule Conditions”. In: *Proceedings of the Eighth International Conference on Data Engineering, February 3-7, 1992, Tempe, Arizona, USA*. Edited by F. Golshani. IEEE Computer Society, 1992, pages 88–97. DOI: 10.1109/ICDE.1992.213202.

[58] K. Yi, H. Yu, J. Yang, G. Xia, and Y. Chen. “Efficient Maintenance of Materialized Top-k Views”. In: *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*. Edited by U. Dayal, K. Ramamritham, and T. M. Vijayaraman. IEEE Computer Society, 2003, pages 189–200. DOI: 10.1109/ICDE.2003.1260792.

Datalyzer
A web platform for streaming applications

Mario González-Jiménez and Juan de Lara

miso research group
Universidad Autónoma de Madrid (Spain)
Mario.GonzalezJ@uam.es, Juan.deLara@uam.es

Nowadays, streaming data are continuously generated from thousands of sources, including social networks, mobile apps, sensors, and many more. Hence, it becomes very useful to build applications able to process these data, with the purpose of filtering interesting parts, and monitor their runtime evolution. In this report we describe Datalyzer, an approach to create streaming data applications on the cloud based on a visual language. Datalyzer provides a facility to describe streaming data sources in an open way, and a visual language to describe the execution flow of the streaming application. In this report, we describe how the HPI SOC Lab infrastructure has helped us to develop and test Datalyzer and discuss oportunities for further developments.

1 Introduction

We live in a hyper-connected society, where 2.5 quintillions bytes of data are created every day.[1] Millions of heterogeneous sources around the world produce continuous streams of data, including sensors, social networks, IoT devices, mobile and web applications, among many others. Therefore, there is an increasing need to observe, monitor, analyse, detect anomalies [5] and create apps out of these heterogeneous data. However, currently, building such applications is only possible by engineers with specialized technical knowledge, and having expensive hardware at their disposal. Moreover, the required technologies for the demanding requirements of this kind of applications tend to have a steep learning curve and can be difficult to install, configure, and run [1].

In order to solve these issues, we are developing Datalyzer, a cloud-based platform to build streaming data applications. Our approach follows model-based development principles [2] and is based on a domain-specific visual language (DSL) to facilitate programming by non-experts. Datalyzer is able to synthesize code—based on Apache Kafka[2]—compile it, configure the application, and run it in an automated way. This way, the end user uses the web browser to visually create the model of the application to be built. As the approach is cloud-based, there is no need for difficult

[1] https://www.domo.com/learn/data-never-sleeps-5 (last accessed 2018-04-01).
[2] https://kafka.apache.org/ (last accessed 2018-04-01).

configurations by the end user. Moreover, the approach is extensible, and relies on a common format to describe live data sources. This permits the addition of new heterogeneous sources as needed, which can then be used in Datalyzer programs.

In this report, we describe the main principles of Datalyzer (based on [4]), findings about the application using HPI Future SOC LAB resources and next steps proposed for our project. Please note that we have used the HPI FutureSOC Lab resources for two periods (Fall 2017 and Spring 2018). More information on Datalyzer can be found at [4].

2 Datalyzer

This section describes the main ingredients of Datalyzer: a uniform, extensible representation of streaming data sources (section 2.1) and a visual DSL to describe execution workflows (section 2.2).

2.1 Data Sources

Dynamic data sources available on the web are heterogeneous with respect to the protocols and technologies used, and their access architectures, typically either pull-based (polling) or push-based (streaming) [3]. Our goal is to be able to create dynamic data streams out of any source type, and hence our approach permits abstracting away the technical details of the different sources (e.g. whether pull or push). This way, we can use them as dynamic live data channels in a uniform way. For this purpose, we have desinged a declarative description model for data sources, which facilitates the incorporation of new sources.

A scheme of the description model is shown in figure 1. DataSourceType describes different types of data transmission technologies like TCP Socket or REST, but also non-standard technologies like Satori[3] or Twitter.[4] Once a DataSourceType (e.g. REST) is described, it is possible to describe DataSources of such type (e.g. OpenWeatherMap).

2.2 DSL

Once the data sources of interest are defined, the user can create her application using a visual DSL we have designed. In practice, this creation process occurs within a web browser, using a drag and drop mechanism. After finishing the description of the application, a code generator we have built synthesizes the program code, compiles it, and finally executes the described streaming application. By using the DSL, the user can instantiate data sources among those previously defined.

[3]https://www.satori.com/ (last accessed 2018-04-01).
[4]https://developer.twitter.com/ (last accessed 2018-04-01).

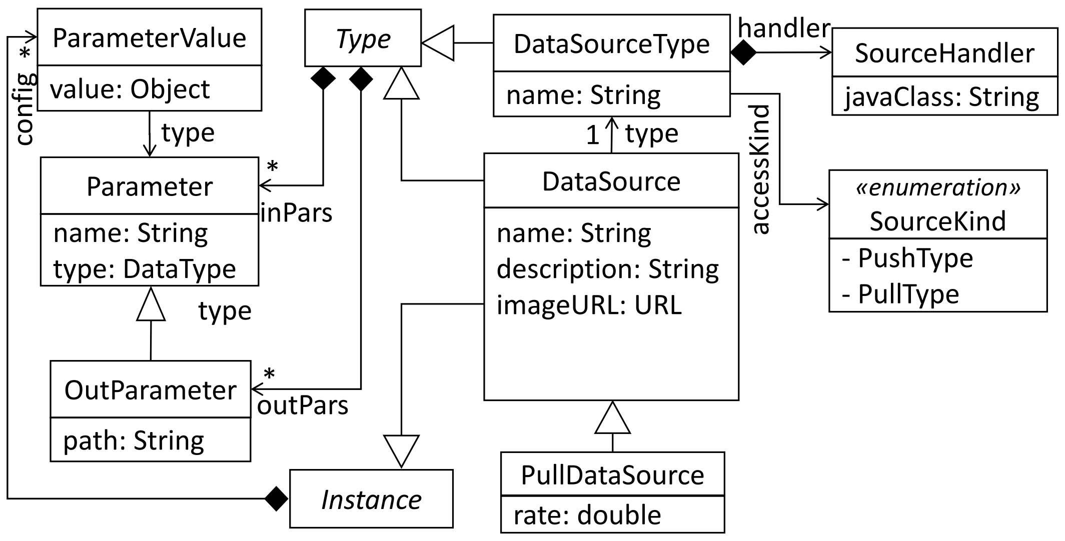


Figure 1: Conceptual model (excerpt) for dynamic data sources

For example, if she defined a TCP socket data source (data source type: socket) and a OpenWeatherMap[5] data source (data source type: REST), she can instantiate and use them in the application she is creating. When a data source is added to the application, the user needs to configure input parameters defined in the model (figure 1) as a specific port for a TCP socket or a city name for OpenWeatherMap. Each data source must be connected with a dataflow pipeline where the data will be received. Over a pipeline, the user can apply transformations such as filtering, selecting values of interest or joining data from different data sources. A pipeline can be connected to other pipelines, but must be connected to at least one terminal node which produces a visualization for the data, a trigger, or persists the data. There are different types of visualizations for data that a user can select, such as tables or dynamic bar and line charts. In the future, we want to implement an output streaming data channel in order to make it possible to use Datalyzer as a service in real-time for external applications. An example of a simple application created with DSL is shown in figure 2. When the user has described her application including data sources, pipelines and terminal nodes, she can compile it and run it by accessing to the dashboard panel which is explained in the next section.

3 Architecture

Datalyzer is composed of several components and technologies. In order to understand our goal when using the HPI FutureSOC Lab infrastructure, the difficulties of the project and complexity of deploying it on the cloud, this section gives an

[5]https://openweathermap.org/api (last accessed 2018-04-01).

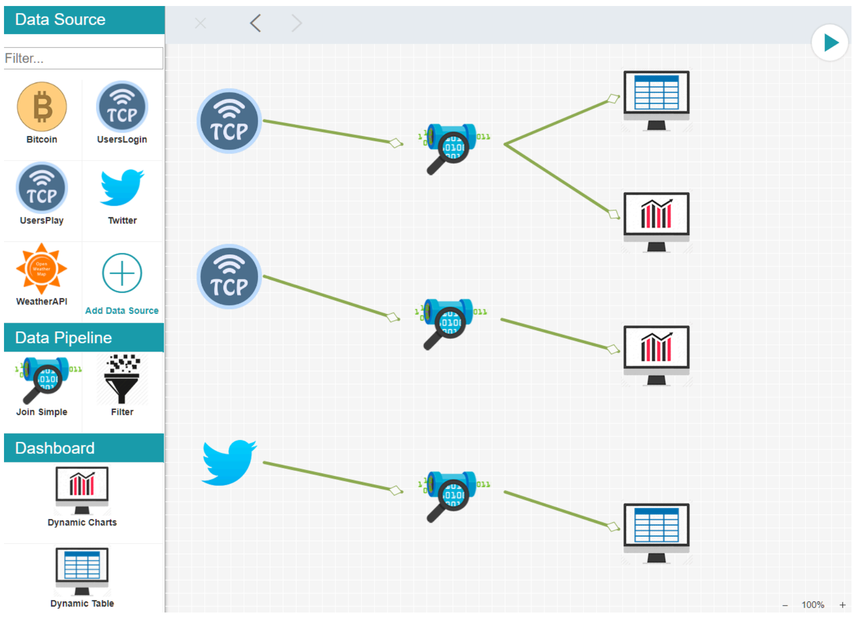


Figure 2: An example application created in the DSL with three data sources, three pipelines and several chart visualizations

overview of the architecture of Datalyzer that is shown in figure 3. The components are as follows:

Web Platform A web platform developed using NodeJS and MongoDB (see label 1 in figure 3). In this platform, users may create projects that contain the models of the streaming applications. The platform permits managing these projects, listing them, editing and executing the models.

Data Source Repository The description of data source types and data sources (seen in section 2.1) are stored in a MongoDB database (label 2). This database is connected with the DSL in order to show the different data sources in the editor palette.

DSL The DSL editor is integrated in the web platform (label 3). This editor is accessed whenever a project is edited. The DSL editor is built in Javascript and contains built-in validations to ensure (syntactical) model correctness at design time. If a model is correct, it can be stored in the database so that it can be opened, modified or executed later.

Code Generation We have built a code generator in Java that is invoked by the web server whenever the application is to be executed (label 4). The end result is an independent, fully functional Java project (based on Apache Kafka), which expects to receive and process the live data sources as specified (label 5).

Streaming Application The generated application (label 6) is executed on the sever. It uses Apache Kafka as a basis to create the data channels. This technology offers fast data pipelines, which are fault tolerant, scalable and distributed. The generated code processes these data channels as specified using the DSL. Once a data channel is initialized and opened, it waits to receive information from the data sources. When data is received, these are inserted in the data channel and processed. This continues until the application is paused or stopped.

Dashboard The streaming applications are executed in a server, so that the user does not have direct access to them. To allow interaction, we have developed a graphical, browser-based dashboard, which is integrated in the web platform (see label 7). The developed interface contains controls to monitor the application status (running on/off); management controls to execute, pause and stop the application; and several widgets developed in JavaScript to visualize the data (e.g. table, dynamic chart).

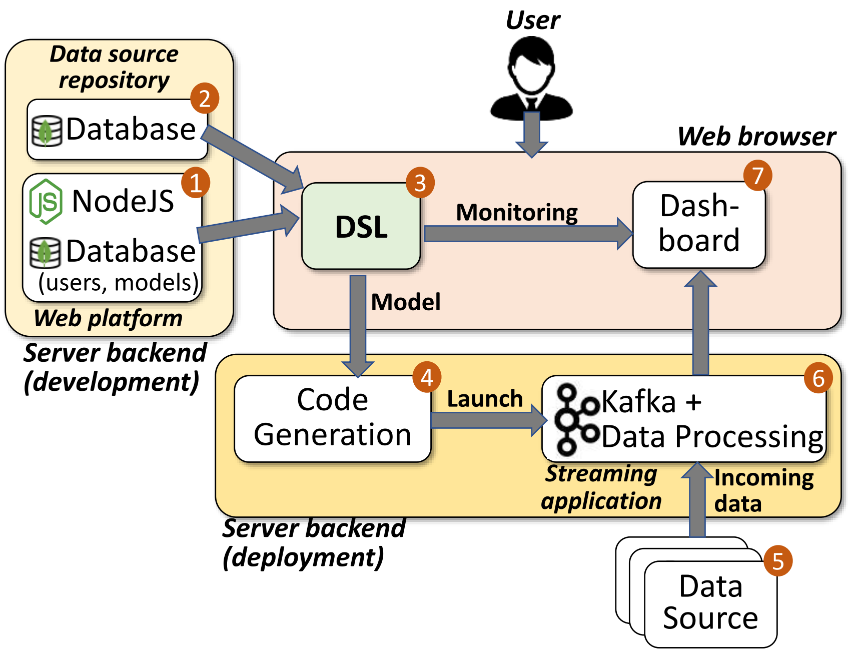


Figure 3: Datalyzer architecture

4 Use of HPI Future SOC Lab resources

We have been assigned 3 virtual machines with Ubuntu 16.04, 8 GB, 4 x Intel Pentium 4 @2.4 GHz cores and 100 GB HDD. Our development environment in our group lacks access to a cloud infrastructure and this testing process is crucial to know the technical limits of Datalyzer. Therefore, we prepared one VM as a cloud pre-production environment to deploy and to test our software. The others VM were required to redesign a distributed architecture in case of detecting bottlenecks in the testing process. However, this has not been necessary as we explain in the next section. We have prepared black-box tests in order to evaluate all features developed. We have also designed white-box tests to detect possible bottlenecks in any component (previous section). Finally, we have evaluated performance, stability and resources consumption by monitoring our application.

4.1 Findings

As we previously explain, this is our second period in HPI Future SOC Lab. Unfortunately, during last period our project had been delayed respect to the project schedule and we could not achieve the goals proposed. Even so, we detected some crucial problems that we have addressed. All technologies and frameworks used in our project are multi-platform, but there are some features that are launched in an autonomous way using scripting. Obviously, these scripts are developed specific for the OS used. In particular, our project has been developed on Windows so we need a version of the scripts prepared for running in Linux or MAC. Further, some technologies that we used like Apache Kafka or MongDB need a specific configuration before our application can be deployed and run.

Overall, our software was not portable and really difficult to deploy and this aspect is a must. For this reason, we added Docker[6] in the project. Docker is a technology that allows us to create a package—called container—with our application and all its dependencies. Containers works in a similar way to a virtual machine, and can be executed independent of the OS using the Docker engine. We have replicated our development environment in a container so now our application is fully portable and can be deployed and executed in seconds (a Docker installation is required). The container executes the same code, configuration and dependencies version independent of the infrastructure where it is running. Therefore, we deployed Datalyzer in our VM assigned and passed successfully most of the black-box tests. We detected some minor bugs that will be fixed in the next phase of the project.

To test the stability of the system we run projects for many hours and analysed the data received and the data produced. To facilitate testing, we developed a mock up dynamic data source, which can be configured with the data velocity. We did not observe significant data loss in thousand of messages received, as well as no delay in the data consumption. About resources usage, the system needs around 500 MB of RAM for the web platform and the Kafka server running on background. For each project created and running on the cloud, the system needs around 100 MB of extra RAM. Please note that the projects we used for testing were simple examples with only one or two pipelines. The container size is 1 GB, so disk space is not an issue. CPU usage is variable, not mainly on the number of projects running but depending on the frequency of the data consumption and the data processing we added in the projects. Performance have worked as we expected as long as there are enough

[6]https://www.docker.com/ (last accessed 2018-04-01).

resources available. Overall, we can conclude that our application is prepared to support several users or projects running at the same time which is our goal. To support hundreds or even thousand of projects, a distribute architecture must be discussed in the future. For example, instead of using a single container with all the system, we may create a cluster with several nodes and a load balance to distribute the resources.

4.2 Next steps

Our next step is to fix the minor bugs detected. Then, we will build a container with the new version of DATALYZER. This version will be completely functional, stable and can be easily executed in any computer so we are prepared to publish it as a open source software in GitHub and a website that we will develop. Moreover, we are working in a case study using public open data. In particular, we want to use datasets about air pollution or traffic (which are updated in real time) in the city of Madrid[7] Our idea is to create projects that can monitor these data and provide information to final users in real time. For example, adding a event capable of sending a message (via email or social media) when a traffic jam is detected, or when traffic restrictions are expected, according to the new city hall regulations.

5 Conclusions

In this report, we have described DATALYZER, a cloud-based approach for the creation of streaming data applications, based on a visual DSL. The approach permits a visual description of the application, which is then compiled, deployed, and executed automatically. The approach is extensible, as new data sources can be added on the fly. Overall, the approach aims at lowering the entry barrier for the creation and execution of streaming data applications.

We have fully deployed DATALYZER on the infrastructure facilitated by the HPI FutureSOC Lab. We have tested it can be executed in a desktop or a cloud infrastructure. There are technical limitations depending on the hardware but DATALYZER should be capable of running several projects at the same time. Finally, we discussed different case study using public open data.

References

[1] M. D. de Assunção, A. D. S. Veith, and R. Buyya. "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions". In: *J. Network and Computer Applications* 103 (2018), pages 1–17.

[7] https://datos.madrid.es/portal/site/egob/ (last accessed 2018-04-01).

[2] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice, Second Edition*. Synthesis Lectures on Sw. Eng. Morgan & Claypool Publishers, 2017.

[3] A. Harth, C. A. Knoblock, S. Stadtmüller, R. Studer, and P. A. Szekely. "On-the-fly Integration of Static and Dynamic Sources". In: *COLD*. Volume 1034. CEUR Workshop Procs. 2013.

[4] M. G. Jiménez and J. de Lara. "Datalyzer: streaming data applications made easy". In: *ICWE*. 2018, in press.

[5] L. Rettig, M. Khayati, P. Cudré-Mauroux, and M. Piórkowski. "Online anomaly detection over Big Data streams". In: *2015 IEEE Int. Conf. on Big Data*. IEEE, 2015, pages 1113–1122.

Hybrid Modelling of Aluminium Smelting Bath Chemistry Process

An approach combining SAP HANA in-memory processing, machine learning algorithms and mathematical theoretical models

Fábio M. Soares and Roberto C. L. De Oliveira

Post-Graduation Program in Electrical Engineering
Federal University of Pará, Brazil
fms@ufpa.br, limao@ufpa.br

Aluminium smelting processes are typically complex and multidisciplinary, hard to be modeled and controlled. Recent advances in technology, such as IIoT and the Industrie 4.0 paradigm, have helped this industry to overcome some of these challenges by providing rich information, however good hardware capable of processing a great amount of data are often available on the cloud, whilst most of industries' plant floor have limited hardware and Internet connectivity. In this sense technologies like in-memory and parallel computing offer an increased performance for processing huge amounts of data, thereby helping in process modeling and subsequent early detection of irregularities. Plant floor systems would be provided with the result of such processing from the cloud in order to ensure good production. This report show the results of this approach in the modelling of bath chemistry process variables, whereby we simulate some situations to check how accurate the forecasts are in the n-step ahead predictions.

1 Project Idea

Aluminium smelting involves many fields within chemistry and physics, and in a higher level, economics, i.e. it is multidisciplinary [4]. This is reflected in the huge databases this industry produces every day, making nearly impossible to manage without technological tools such as automation. The production process itself faces many challenges that prevent the plant to reach its maximum level of productivity. Process modelling helps in planning productivity by forecasting trends and simulating alternative scenarios. Most of the modelling being made on Aluminium Smelting is based on the physical laws of the process [1, 2], which are too complex. Data science techniques, on the other hand enable a shortcut to build a model based on the process dynamics itself [3, 5]. While each approach has its flaws separately, they can complement each other when applied in combination, as differential equations can be solved and used to add more data for data science algorithms to work on.

1.1 Approach used

In the last periods' report, we have solved differential equations to generate data that have been added into the process database, including variables that are not directly measured and estimating the values of variables in higher sampling frequency. We expanded the original database by applying 20 mathematical models to add variables that are absent in the database, and resampled the data from 24–64 hours to 4-hour sampling, as shown schematically in Figure 1. Taking all this number of variables a data for every single production unit, which in the aluminium smelting is called cell or pot, we behold a really huge database. The objective here is to predict the future state of all the production cells, i.e. a model is needed. However, due to the physical nature of the cells, it is not guaranteed that all cells will behave similarly, in the spite of sharing the same construction technology. In addition, even for advanced hardware and software technology, a too huge database implies in a long time to process and build models.

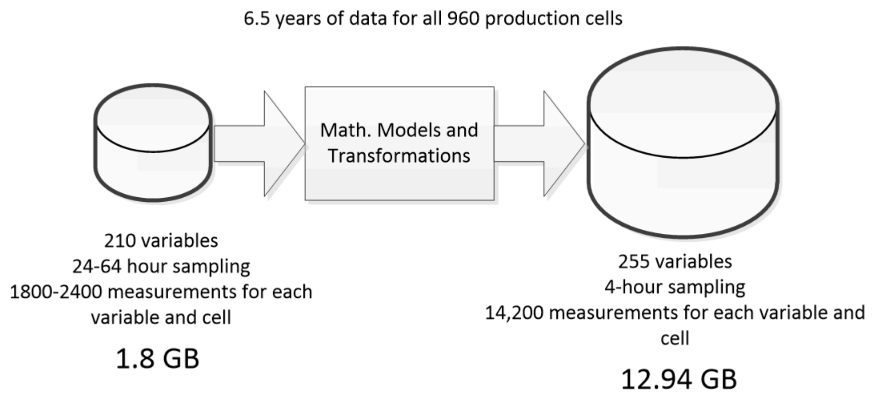

6.5 years of data for all 960 production cells
Math. Models and Transformations
210 variables
24-64 hour sampling
1800-2400 measurements for each variable and cell
1.8 GB
255 variables
4-hour sampling
14,200 measurements for each variable and cell
12.94 GB

Figure 1: Database Expansion

1.2 Clustering and Preprocessing

Lima et al. [6] performed a study on clustering aluminium reduction cells using statistical measures such as mean, median and standard deviation, as suggested in the work of Horvath and Vircikova [5]. We used the same approach to find clusters of cells whose data will be applied to build models for each cluster separately, assuming all the cells in the cluster share close behaviors. In addition, as we should take into account lags for each variable, we apply dimensionality reduction by means of Principal Components Analysis, to eliminate eventual redundancy in the data. In this report we compare the models we already have with the models obtained using data from clusters and using PCA.

2 Used HPI FutureSOC Lab resources

Our project used one instance of SAP HANA database server to store the data in a column-stored table, enabling fast aggregate operations, such as mean and sum. Also we loaded our models into 8 Virtual Machines; 4 of them are extra large to process the cluster models for each single cell from each of the 4 potlines, and the other 4 are only large to process and potline related data using the general models. The mathematical models that generate lump paramater data run on both extra large and large virtual machines, while the distributed parameter mathematical models, which generate spatial and time data for each variable, are run on the multicore server and the tesla Nvidia card. However the boundary conditions to generate these data couldn't be determined, as this depends also on measurements to be taken from the potlines.

3 Findings

In the previous periods we have built simple models with the least possible number of hidden neurons in Multilayer Perceptron Regressors. For this period we wanted to attest: the capability of the models to predict in a long run (multi-step prediction); and to check the performance when applying PCA and clustering to reduce the training size. This is particularly important because in a industrial environment the plant data may suffer from biased measurements or even phenomena that are not yet fully covered in the traditional models, causing the data generated to be very different from those used in the training phase. On the other hand, if the plant dynamics is changed and this is reflected in the data, a runtime retraining of the model may improve the predictions. Table 1 shows the results for two variables already modeled comparing the test Mean Squared Error (MSE) performed by the last model (general model) vs. cluster-based models.

Table 1: MSE Errors comparing cluster based and general model

	General based model		Cluster based model	
Steps	Liquidus Temp.	Current Efficiency	Liquidus Temp.	Current Efficiency
1	0.029 112	0.048 33	0.005 89	0.008 680 1
2	0.093 07	0.102 647	0.028 627 5	0.014 583 9
3	0.170 59	0.231 983	0.065 151 9	0.086 620 3

It is worth noting that the MSE is calculated over the normalized output value, subtracting the mean and diving by the standard deviation. The predictions get worse after the 2nd step (8 hours ahead).

3.1 Effect of PCA

By using PCA, we checked that about 20 components explain more than 99 percent of the data variance. Therefore we applied the PCA on each of the models to see if the results are improved. The scatter and line charts show the samples for 1, 2 and 3 step ahead prediction in comparison with the previous models. PCA doesn't seem to improve the results for the short term, but when compared with multi-step the results seem to be more accurate, what can be explained by a lesser effect of overtraining.

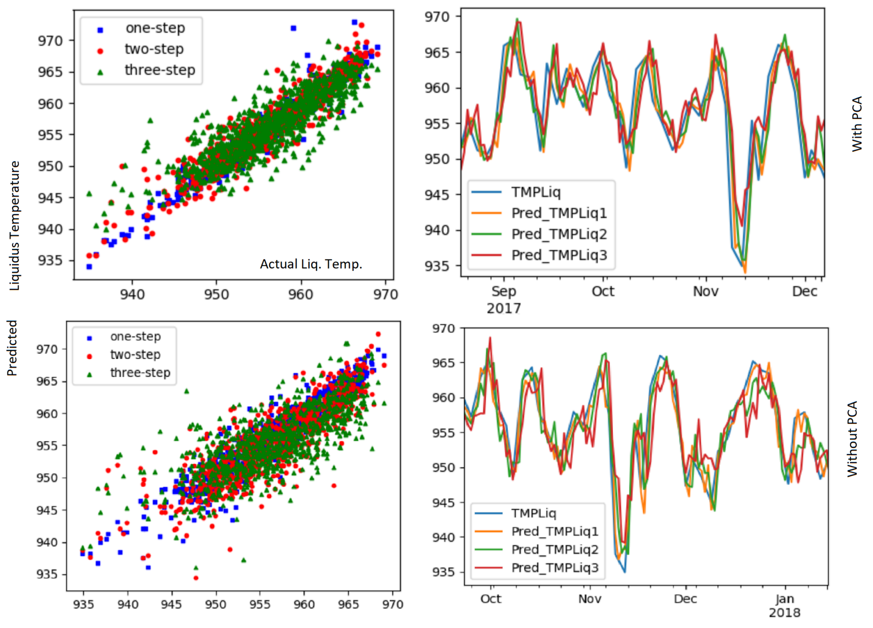
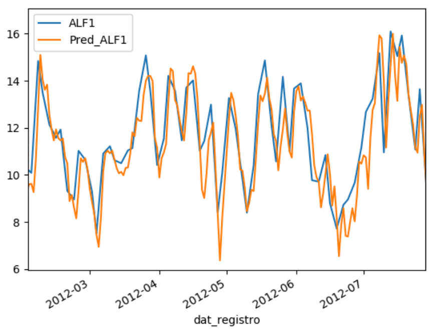


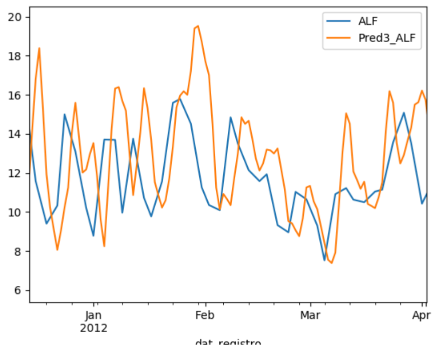
Figure 2: PCA with cluster based model

3.2 Retraining

We chose one particular situation (Aluminium Fluoride analysis) to test the retraining of a model. This variable is very sensitive to errors, as the concentration may significantly vary, depending on where it is measured. From a specific period of time to another there was a change of behavior in a given cell, whether biased by the process or operation. For every week the model was retrained to be updated included the last data. For this case, using PCA and cluster-based information, the results are for one-step prediction, while for a longer term, the predictions differ a lot from the reality.



Retraining with 1 step prediction



Retraining with 3-steps prediction

Figure 3: Retraining with PCA and cluster based

4 Discussion and next steps

The results show that dimensionality reduction and clustering to reduce training database seem to improve much for the one-step prediction, what can be explained by the fact that much of the future information are already present in the delayed output variable. On the other hand, multi-step predictions, that imply a simulation of the plant in feedback, loses accuracy the longer the prediction window. Many studies on the electrolysis cells have shown that the cell dynamics is subject to change from time to time, therefore it is very unlikely for a model to be built once and run accurately forever in production environment. This calls for a continuous retraining, to include newer information. Retraining indeed has helped in the one-step prediction, however for longer terms, the predictions can't be so accurate, what can be explained by a number of reasons, including future disturbances that impact on the pot faster than other variables. Next steps should include spatial data and seek ways to reduce the training time, without losing accuracy. Spatial data variations in addition to time delayed variables may add good information to improve the accuracy for multi-step ahead predictions. Also it is worth trying to build models with less components, speeding up the time to retrain a model with recent and new data.

References

[1] C. A. P. Braga and J. V. da Fonseca Netto. "A dynamic state observer to control the energy consumption in aluminium production cells". In: *Systems Science & Control Engineering* 4.1 (2016), pages 307–319. DOI: 10.1080/21642583.2016.1238325.

[2] M. Dupuis. "Essential Readings in Light Metals". In: edited by B. G., D. M, and T. G. Springer, Cham, 2016. Chapter Mathematical Modelling of Aliumnium Reduction Cell Potshell Deformation. ISBN: 978-3-319-48156-2.

[3] L. Fortuna et al. *soft Sensors for Monitoring and Control of Industrial Processes*. London, UK: Aluminium Verlag, 2007. ISBN: 978-1-84628-480-9.

[4] K. Grjotheim and H. Kvande. *Introduction to Aluminium Electrolysis*. Düsseldorf, Germany: Aluminium Verlag, 1993. ISBN: 978-3-410-22027-5.

[5] M. Horvath and E. Vircikova. "Data Mining Model for Quality Control of Primary Aluminium Production Process". In: *Management and Production Engineering Review* 3.4 (2012), pages 47–53. DOI: 10.2478/v10270-012-0033-x.

[6] F. A. N. de Lima, A. M. F. de Souza, F. M. Soares, D. L. Cardoso, and R. C. L. de Oliveira. "Clustering Aluminum Smelting Potlines Using Fuzzy C-Means and K-Means Algorithms". In: *Light Metals 2017*. Edited by A. P. Ratvik. Cham: Springer International Publishing, 2017, pages 589–597. ISBN: 978-3-319-51541-0. DOI: 10.1007/978-3-319-51541-0_73.

Benchmarking Graph Business Intelligence workloads

Gábor Szárnyas[1], József Marton[2], János Benjamin Antal, and Dániel Varró[1]

[1] MTA-BME Lendület Cyber-Physical Systems Research Group
Department of Measurement and Information Systems
Budapest University of Technology and Economics
szarnyas@mit.bme.hu, varro@mit.bme.hu
[2] Database Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
marton@db.bme.hu

In this report, we present our preliminary results on the early version of the LDBC Social Network Benchmark's Business Intelligence (BI) workload. This workload tests graph data management systems on a graph business analytical workload. Its queries involve complex aggregations and navigations (joins) that touch large data volumes, which is typical in BI workloads, yet they depend heavily on graph functionality such as connectivity tests and path finding. We outline the motivation for this new benchmark, which we derived from many interactions with the graph database industry and its users, and situate it in a scenario of social network analysis.

1 Introduction

The explosion of network- or graph-shaped data has increased the demand for tools analyzing such data sets, including specialized cluster framework APIs, SQL extensions or graph databases. The diversity of tools and technologies creates the need for standard benchmarks that help practitioners identify the technologies that suit their application needs. Additionally, benchmarks spur competition among vendors and stimulate research in the field, as TPC benchmarks [7, 17, 19] have done for the RDBMS industry during the past 20 years.

Benchmarks with aggregation-heavy OLAP-like Business Intelligence (BI) workloads on graphs are still a rather unexplored area, and existing proposals do not fully capture the complex nature of such workloads. Currently, the only benchmark with global queries and aggregations on graph-like data is the Berlin SPARQL Benchmark's BI use case [6]. However, while proposed on RDF, it is exactly equivalent to and exists in a SQL variation on flat tables in a star schema, i.e. its dataset lacks a true graph structure and its queries thus do not require graph functionality.

Graph BI workloads differ from other types of graph query workloads in that large portions of the graph are explored in search of occurrences of graph patterns. Compared to graph analytics workloads, the patterns under search combine both structural and attribute predicates of varying complexity [20], from basic graph patterns [3] to more complex unbound patterns that rely on different reachability semantics (e.g. paths, trails). The identified patterns are typically grouped, aggregated, and sorted to summarize the results, which are used to assist the user in critical decision making.

BI workloads on graphs are particularly challenging because they usually lead to large search spaces and consequently, to large intermediate results. Thus, systems that are not prepared to efficiently prune the search space—by finding good graph traversal orderings, leveraging reachability indexes, or taking advantage of top-k semantics to progressively reduce the size of candidate results—are heavily penalized. Moreover, the complex structure of real graphs induces difficult-to-predict, scattered memory access patterns, which can limit the memory bandwidth saturation by orders of magnitude if computations are not arranged correctly [22]. Finally, some complex graph patterns become difficult to express even with the most advanced query languages, leading to large and verbose queries, which are difficult to write and maintain.

The LDBC Social Network Benchmark's Business Intelligence workload was presented at the GRADES-NDA workshop of SIGMOD 2018. This report is based on workshop paper [23] (which provides a look into the design of the BI workload) along with our previous work [24] (which collects related graph benchmarks).

2 Implementation and Evaluation

We implemented the Business Intelligence workload in the LDBC SNB framework. The current benchmark ecosystem consists of multiple components, most importantly the specification,[3] the data generator,[4] the driver,[5] and the implementations.[6]

Interactive workload We reworked the implementation framework and queries of the *Interactive workload*. The definitive publications on the workload [8] only presented a preliminary evaluation of this workload: an imperative (C++) implementation executed in the *Sparksee* embedded graph database [15] and a relational (SQL-based) *Virtuoso* implementation [9].

Business Intelligence workload We implemented the queries of the BI workload for PostgreSQL and Sparksee. Additionally, a partial PGX implementation was contributed by Oracle Labs [24].

Highlights of the benchmark results Figure 1 and figure 2 show benchmark results of the LDBC SNB's *Interactive* and *Business Intelligence* workloads, respectively, on scale factors SF1, SF3, and SF10. For all scale factors, the SQL implementation of the

[3]https://github.com/ldbc/ldbc_snb_docs (last accessed 2018-04-01).
[4]https://github.com/ldbc/ldbc_snb_datagen (last accessed 2018-04-01).
[5]https://github.com/ldbc/ldbc_snb_driver (last accessed 2018-04-01).
[6]https://github.com/ldbc/ldbc_snb_implementations (last accessed 2018-04-01).

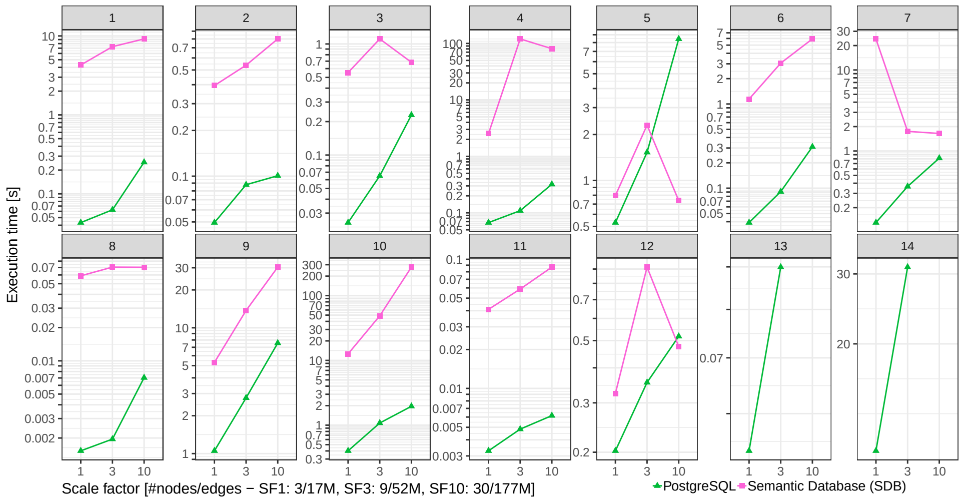


Figure 1: Execution times for the complex queries of the SNB Interactive workload

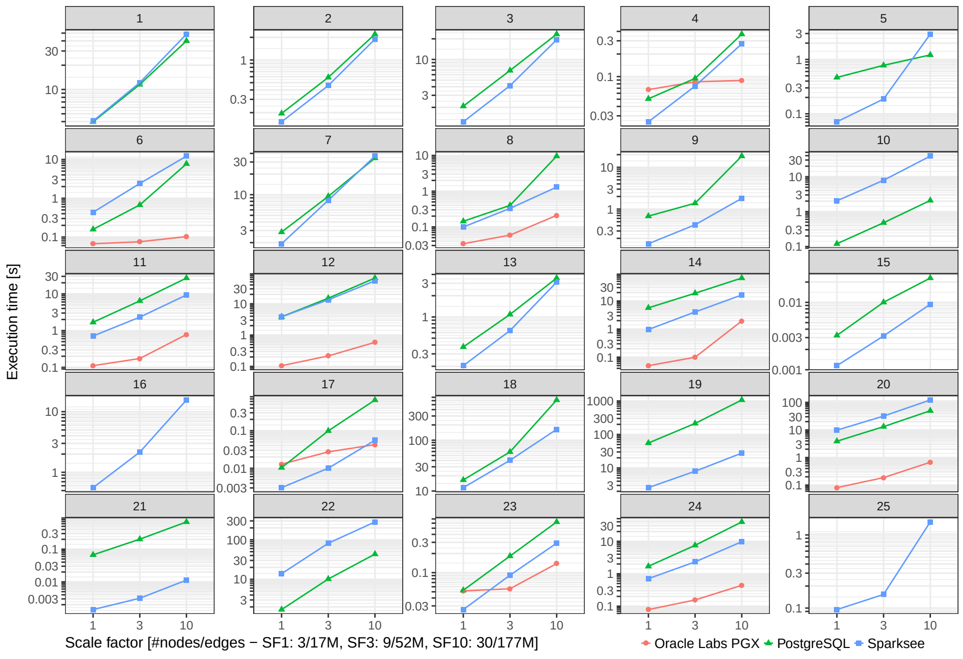


Figure 2: Execution times for queries of the SNB Business Intelligence workload

Interactive queries 1–4, 6–8, and 11–12 can be evaluated in sub-seconds, while queries 5, 9, and 10 take less than 10 seconds. The most challenging part of this workload is clearly query 14, which takes more than half a minute for SF3 and cannot be evaluated on SF10.

It is easy to see that the *BI workload* is significantly more complex than *Interactive*: about half of the 25 queries take more than 10 seconds (1, 3, 7, 8, 9, 11, 12, 14, 18, 19, 20, 22, and 24) for SF10. The most challenging part of this workload are queries 16 and 25, which cannot be evaluated for even the smallest SF.

3 Related Work

There are several well-defined performance benchmarks for assessing the performance of RDF and graph database technologies. We present a comparison of available benchmarks in table 1.

One of the first ontology benchmarks are the *Lehigh University Benchmark* (*LUBM*) [12], and its improved version, the *UOBM Ontology Benchmark* [14]. These are tailored to measure reasoning capabilities of ontology reasoners. Another early benchmark used the *Barton dataset* [1] for benchmarking RDF stores. The benchmark simulates a user browsing through the RDF Barton online catalog. Originally, the queries were formulated in SQL, but they can be adapted to SPARQL as well.

SP²Bench [21] is a SPARQL benchmark that measures the execution time of various queries. The goal of this benchmark is to measure the query evaluation performance of different tools for a single set of SPARQL queries that contain most language elements. The artificially generated data is based on the real-world DBLP bibliography; this way instance models of different sizes reflect the structure and complexity of the original real-world dataset.

The *Berlin SPARQL Benchmark* (*BSBM*) [6] measures SPARQL query evaluation throughput for an e-commerce case study modeled in RDF. The benchmark uses a single dataset, but recognizes several use cases with their own set of queries. The dataset scales in model size (10 million–150 billion), but does not vary in structure.

In the *SPLODGE* [11] benchmark, SPARQL queries are generated systematically, based on metrics for a predefined dataset. The method supports distributed SPARQL queries (via the SERVICE keyword), however the implementation scales only up to three steps of navigation, due to the resource consumption of the generator.

The *DBpedia SPARQL benchmark* [16] presents a general SPARQL benchmark procedure, applied to the DBpedia knowledge base. The benchmark is based on query-log mining, clustering and SPARQL feature analysis.

LinkBench [4] was developed in collaboration with Facebook and is based on its social graph. It tests 10 operations, including reads and updates, such as object_get, assoc_multiget, and assoc_update. It measures both the *latency* and the *throughput* of the system under benchmark. *WatDiv* [2] workload generator that allows users to fine-tune the *structuredness* of the synthesized graphs and the complexity of generated queries. *Stream WatDiv* [10] adds streaming updates to the workload, and equips SPARQL queries with time windows (using languages such as CQELS [18]

Table 1: Benchmarks for semantic and graph databases. Notations—*Largest graph* column: “M” stands for million, “B” stands for billion; *#queries* column: ⊛ the benchmark allows users to generate an arbitrary number of queries; *Focus metric* column: *thr.* = throughput; *Updates* column: ⊗ measuring the performance of updates is an important aspect of the benchmark, ⊘ the benchmark uses updates, but the performance of reevaluation after updates is not an important aspect, ○ the benchmark does not consider updates.

Benchmark	Ref.	Graphs	Largest graph	#classes	#predicates	#queries	Multiple users	Workload profile	Focus metric	Updates
LUBM	[12]	synthetic	6.9M	43	32	14	○	analyzing university data	inferencing performance	○
Barton	[1]	real	50M	11	28	7	○	library search	response time	○
SP^{2}Bench	[21]	synthetic	1B+	8	22	12	○	publication research	response time	○
BSBM	[6]	synthetic	150B	8	51	12	⊗	e-commerce	throughput	⊘
DBpedia	[16]	real	300M	8	1200	25	○	queries on DBpedia	throughput	○
LinkBench	[4]	synthetic	1B+	25+	100+	10+	○	social network	latency/thr.	⊗
WatDiv	[2]	synthetic	10B+	17	85	⊛	○	e-commerce	response time	○
SNB Interactive	[8]	synthetic	1B+	19	27	14+	○	local queries on a social network	response time	⊗
LDBC SPB	[13]	synthetic	1B+	74	117	12	○	creative works	response time	⊗
Train Benchmark	[23]	synthetic	23M+	9	13	6	○	validation of models	query/update response time	⊗
SNB BI	[24]	synthetic	1B+	19	27	25	○	BI on a social network	response time	○
Stream WatDiv	[10]	synthetic	10B+	17	85	⊛	○	user activity stream	response time	⊗

and C-SPARQL [5]). The Linked Data Benchmark Council recently published the *Semantic Publishing Benchmark* (SPB) [13] based on a scenario of the BBC media organization. SPB combines a query workload with a stream of update operations.

In the field of model-driven engineering, the Train Benchmark [23] defines a model validation scenario, where a graph model is checked for errors with complex queries, while continuously being updated. The Viatra CPS Benchmark[7] measures the performance of incremental model transformations, presented in a *cyber-physical system allocation* scenario.

4 Conclusion

Our work so far resulted in an early publication describing the LDBC SNB's new BI workload. [24]. Our future plans focus on extending the benchmark specification with updates that remove certain nodes and edges from the graph. The importance of such operations is increasingly recognized with the recent implementation of the *General Data Protection Regulation* (GDPR) [25]. This regulation (superficially) requires organizations collecting user data to provide means that allow users to permanently remove their data from the system, which may prove to be challenging for many database management systems.

References

[1] D. J. Abadi, A. Marcus, S. Madden, and K. Hollenbach. *Using the Barton Libraries Dataset as an RDF Benchmark.* Technical report MIT-CSAIL-TR-2007036. 2007.

[2] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. "Diversified Stress Testing of RDF Data Management Systems". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I.* Edited by P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. Groth, N. F. Noy, K. Janowicz, and C. A. Goble. Volume 8796. Lecture Notes in Computer Science. Springer, 2014, pages 197–212. doi: 10.1007/978-3-319-11964-9_13.

[3] R. Angles, M. Arenas, P. Barceló, P. A. Boncz, G. H. L. Fletcher, C. Gutiérrez, T. Lindaaker, M. Paradies, S. Plantikow, J. F. Sequeda, O. van Rest, and H. Voigt. "G-CORE: A Core for Future Graph Query Languages". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018.* Edited by G. Das, C. M. Jermaine, and P. A. Bernstein. ACM, 2018, pages 1421–1432. doi: 10.1145/3183713.3190654.

[7] https://github.com/viatra/viatra-cps-benchmark (last accessed 2018-04-01).

[4] T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. "LinkBench: a database benchmark based on the Facebook social graph". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. Edited by K. A. Ross, D. Srivastava, and D. Papadias. ACM, 2013, pages 1185–1196. DOI: 10.1145/2463676.2465296.

[5] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. "C-SPARQL: a Continuous Query Language for RDF Data Streams". In: *Int. J. Semantic Comput.* 4.1 (2010), pages 3–25. DOI: 10.1142/S1793351X10000936.

[6] C. Bizer and A. Schultz. "The Berlin SPARQL Benchmark". In: *Int. J. Semantic Web Inf. Syst.* 5.2 (2009), pages 1–24. DOI: 10.4018/jswis.2009040101.

[7] P. A. Boncz, T. Neumann, and O. Erling. "TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark". In: *Performance Characterization and Benchmarking - 5th TPC Technology Conference, TPCTC 2013, Trento, Italy, August 26, 2013, Revised Selected Papers*. Edited by R. Nambiar and M. Poess. Volume 8391. Lecture Notes in Computer Science. Springer, 2013, pages 61–76. DOI: 10.1007/978-3-319-04936-6_5.

[8] O. Erling, A. Averbuch, J. L. Larriba-Pey, H. Chafi, A. Gubichev, A. Prat-Pérez, M.-D. Pham, and P. A. Boncz. "The LDBC Social Network Benchmark: Interactive Workload". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. Edited by T. K. Sellis, S. B. Davidson, and Z. G. Ives. ACM, 2015, pages 619–630. DOI: 10.1145/2723372.2742786.

[9] O. Erling and I. Mikhailov. "RDF Support in the Virtuoso DBMS". In: *Networked Knowledge - Networked Media - Integrating Knowledge Management, New Media Technologies and Semantic Systems*. Edited by T. Pellegrini, S. Auer, K. Tochtermann, and S. Schaffert. Volume 221. Studies in Computational Intelligence. 2009, pages 7–24. DOI: 10.1007/978-3-642-02184-8_2.

[10] L. Gao, L. Golab, M. T. Özsu, and G. Aluç. "Stream WatDiv: A Streaming RDF Benchmark". In: *Proceedings of the International Workshop on Semantic Big Data, SBD@SIGMOD 2018, Houston, TX, USA, June 10, 2018*. Edited by S. Groppe and L. Gruenwald. ACM, 2018, 3:1–3:6. DOI: 10.1145/3208352.3208355.

[11] O. Görlitz, M. Thimm, and S. Staab. "SPLODGE: Systematic Generation of SPARQL Benchmark Queries for Linked Open Data". In: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*. Edited by P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, and E. Blomqvist. Volume 7649. Lecture Notes in Computer Science. Springer, 2012, pages 116–132. DOI: 10.1007/978-3-642-35176-1_8.

[12] Y. Guo, Z. Pan, and J. Heflin. "LUBM: A benchmark for OWL knowledge base systems". In: *J. Web Semant.* 3.2-3 (2005), pages 158–182. DOI: 10.1016/j.websem.2005.06.005.

[13] V. Kotsev, N. Minadakis, V. Papakonstantinou, O. Erling, I. Fundulaki, and A. Kiryakov. "Benchmarking RDF Query Engines: The LDBC Semantic Publishing Benchmark". In: *Proceedings of the Workshop on Benchmarking Linked Data (BLINK 2016) co-located with the 15th International Semantic Web Conference (ISWC), Kobe, Japan, October 18, 2016*. Edited by I. Fundulaki, A. Krithara, A.-C. N. Ngomo, and V. Rentoumi. Volume 1700. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1700/paper-01.pdf.

[14] L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, and S. Liu. "Towards a Complete OWL Ontology Benchmark". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. Edited by Y. Sure and J. Domingue. Volume 4011. Lecture Notes in Computer Science. Springer, 2006, pages 125–139. DOI: 10.1007/11762256_12.

[15] N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J. L. Larriba-Pey. "Dex: high-performance exploration on large graphs for information retrieval". In: *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*. Edited by M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão. ACM, 2007, pages 573–582. DOI: 10.1145/1321440.1321521.

[16] M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. "DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data". In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*. Edited by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, and E. Blomqvist. Volume 7031. Lecture Notes in Computer Science. Springer, 2011, pages 454–469. DOI: 10.1007/978-3-642-25073-6_29.

[17] R. O. Nambiar and M. Poess. "The Making of TPC-DS". In: *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. Edited by U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim. ACM, 2006, pages 1049–1058. URL: http://dl.acm.org/citation.cfm?id=1164217.

[18] D. L. Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. "A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data". In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*. Edited by L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, and E. Blomqvist. Volume 7031. Lecture Notes in Computer Science. Springer, 2011, pages 370–388. DOI: 10.1007/978-3-642-25073-6_24.

[19] M. Pöss and C. Floyd. "New TPC Benchmarks for Decision Support and Web Commerce". In: *SIGMOD Rec.* 29.4 (2000), pages 64–71. DOI: 10.1145/369275.369291.

[20] S. Sakr, S. Elnikety, and Y. He. "G-SPARQL: a hybrid engine for querying large attributed graphs". In: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. Edited by X.-w. Chen, G. Lebanon, H. Wang, and M. J. Zaki. ACM, 2012, pages 335–344. DOI: 10.1145/2396761.2396806.

[21] M. Schmidt, T. Hornung, M. Meier, C. Pinkel, and G. Lausen. "SP^{2}Bench: A SPARQL Performance Benchmark". In: *Semantic Web Information Management - A Model-Based Perspective*. Edited by R. D. Virgilio, F. Giunchiglia, and L. Tanca. Springer, 2009, pages 371–393. DOI: 10.1007/978-3-642-04329-1_16.

[22] B. Shao, Y. Li, H. Wang, and H. Xia. "Trinity Graph Engine and its Applications". In: *IEEE Data Eng. Bull.* 40.3 (2017), pages 18–29. URL: http://sites.computer.org/debull/A17sept/p18.pdf.

[23] G. Szárnyas, B. Izsó, I. Ráth, and D. Varró. "The Train Benchmark: cross-technology performance evaluation of continuous model queries". In: *Softw. Syst. Model.* 17.4 (2018), pages 1365–1393. DOI: 10.1007/s10270-016-0571-8.

[24] G. Szárnyas, A. Prat-Pérez, A. Averbuch, J. Marton, M. Paradies, M. Kaufmann, O. Erling, P. A. Boncz, V. Haprian, and J. B. Antal. "An early look at the LDBC social network benchmark's business intelligence workload". In: *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Houston, TX, USA, June 10, 2018*. Edited by A. Arora, A. Bhattacharya, G. H. L. Fletcher, J. L. Larriba-Pey, S. Roy, and R. West. ACM, 2018, 9:1–9:11. DOI: 10.1145/3210259.3210268.

[25] C. Tankard. "What the GDPR means for businesses". In: *Netw. Secur.* 2016.6 (2016), pages 5–8. DOI: 10.1016/S1353-4858(16)30056-3.

ISBN 978-3-86956-547-7
ISSN 1613-5652