Christoph Boden*, Tilmann Rabl*, and Volker Markl

The Berlin Big Data Center (BBDC)

Abstract: The last decade has been characterized by the collection and availability of unprecedented amounts of data due to rapidly decreasing storage costs and the omnipresence of sensors and data-producing global online-services. In order to process and analyze this data deluge, novel distributed data processing systems resting on the paradigm of data flow such as Apache Hadoop, Apache Spark, or Apache Flink were built and have been scaled to tens of thousands of machines. However, writing efficient implementations of data analysis programs on these systems requires a deep understanding of systems programming, prohibiting large groups of data scientists and analysts from efficiently using this technology. In this article, we present some of the main achievements of the research carried out by the Berlin Big Data Cente (BBDC). We introduce the two domain-specific languages Emma and LARA, which are deeply embedded in Scala and enable declarative specification and the automatic parallelization of data analysis programs, the PEEL Framework for transparent and reproducible benchmark experiments of distributed data processing systems, approaches to foster the interpretability of machine learning models and finally provide an overview of the challenges to be addressed in the second phase of the BBDC.

Keywords: Apache Flink, BBDC, Emma, LARA, PEEL

ACM CCS: Information systems - Database management system engines - Parallel and distributed DBMSs - MapReduce-based systems

1 Introduction

The last decade was marked by the digitization of virtually all aspects of our daily life and our work. Due to the decline in the price of disk storage and the increasing popularity of cloud storage, businesses, communities, households, and public institutions as well as all the sciences are facing a deluge of digital data. This sparked extensive developments of systems and tools to organize these vast amounts of data and to enable complex analyses. However, applying this technology requires a solid background in programming large-scale distributed systems. In 2014 the Berlin Big Data Center[1] (BBDC) started with the goal to overcome this limitation and to enable large-scale data analysis without requiring deep understanding of distributed systems. The BBDC is a competence center for big data, funded by the German Federal Ministry of Education and Research. The BBDC's principal goal is to develop declarative ways of doing scalable data analysis and machine learning and, thus, empowering data scientists with limited or no background in systems programming to do large-scale data analysis. To achieve this goal, the BBDC unites researchers from various disciplines of computer science, in particular data management and machine learning. In the context of this paper, we present some of the major results and achievements of the research efforts as well as outline research questions that we intend to address in the second phase of the BBDC which will span the next three years.

An important aspect of the BBDC has been the further development of Apache Flink [1], a system which originated from a preceding research project called Stratosphere[2]. Apache Flink was introduced into the BBDC as a basis technology. It is a distributed data analysis system for batch and stream processing workloads. It offers the user a reduced level of complexity through the integration of traditional database concepts such as declarative query languages and automatic query optimization. Flink provides a programming model that provides support of iterative algorithms and complex user-defined functions which simplifies the process of creating data analysis programs in comparison with other technologies. Today, Flink is considered one of the most important and most promising projects within the Apache Big Data Stack and has acquired both a significantly increased number of community as well as numerous well-known new users over the past two years. The most important pioneers of data-driven and digital business models such as Amadeus, Zalando, ResearchGate, Alibaba, Uber or Netflix use Flink as early adopters. We discussed Apache Flink and its application in various research projects in detail in a prior article [15] and will

***Corresponding Author: Christoph Boden, Tilmann Rabl, Volker Markl:** Technische Universität Berlin, FG DIMA and DFKI, Germany

1 http://www.bbdc.berlin

2 http://stratosphere.eu/

thus focus on other research achievements of the BBDC in this contribution.

2 Declarative Data Processing: Emma and LARA

Second generation distributed data flow systems such as Apache Flink [3] or Apache Spark[4] have alleviated the major shortcomings of Apache Hadoop, namely the inability to efficiently execute iterative algorithms and a simplistic programming model lacking essential data transformation operations as second-order primitives such as `join`. Both Spark and Flink are based on a distributed collection type equipped with second-order functions that encapsulate parallelism as a principle programming abstraction [20, 8]. While Apache Spark only supports acyclic dataflows and executes iterative computations by lazily unrolling and evaluating dataflows from a Scala-driven loop, Apache Flink provides a native `iteration` operator which enables Flink's runtime to introduce special feedback edges into the data flow, which allows it to cache loop invariant data but requires the use of a dedicated construct at the API level. In Spark, the programmer has to encode the decision to materialize an intermediate result explicitly with a cache primitive. Both systems allow the user to compose complex data flows using a broad range of transformations on these collections. However, a thorough understanding of the underlying execution model and system internals is still essential in order to write efficient programs. Take the well know example of counting the frequency of words in text documents known as *word count* as an example: a straightforward, general and intuitive implementation in Spark is as follows:

```
val words = Array("Lorem", "ipsum", "dolor")
val wordsRDD = sc.parallelize(words)

val counts = wordsRDD.map(word => (word, 1))
  .groupByKey()
  .map(tuple => (tuple._1, tuple._2.sum))
  .collect()
```

(Note that `tuple._1` and `tuple._2` refer to the first and second element inside a tuple respectively.) However, this implementation strategy turns out to be quite inefficient, as no local pre-aggregation is performed and all key-value pairs have to be shuffled prior to aggregation. Spark provides a special `reduceByKey()` operator for these parallel aggregates in order to avoid a global shuffle of all tuples and rather to combine outputs with a common key on each partition before shuffling the data.

```
val counts = wordsRDD.map(word => (word, 1))
  .reduceByKey(_ + _)
  .collect()
```

There still exists quite a high entry barrier due to the required level of understanding of the underlying execution model when programming these systems. The user code is comparatively hard to maintain and read due to the low-level abstractions used and there are several missed opportunities for automatic optimization of the data flows due to hard-coded physical execution strategies. For example, multi-way joins have to be written as a concatenation of binary join function applications, which prohibits join order optimization being carried out.

In order to address these shortcomings and to provide truly declarative specifications of data analysis programs, BBDC researchers developed *Emma*[5] [2, 3], a domain-specific language (DSL) deeply embedded in Scala which enables parallel collection processing through comprehensions - a declarative syntax akin to SQL. In contrast to Spark and Flink, the core abstraction exposed by Emma is a generic type `DataBag`. To illustrate this concept, consider the aforementioned WordCount example in Emma:

```
@emma.lib
object WordCount {

  def apply(docs: DataBag[String]):
  DataBag[(String, Long)] = {

    val words = for {
      line <- docs
      word <- DataBag[String](line.split("\\W+"))
      if word != ""
    } yield word

    val counts = for {
      group <- words.groupBy(identity)
    } yield (group.key, group.values.size)

    counts
  }
}
```

3 https://flink.apache.org/

4 https://spark.apache.org/

5 http://emma-language.org/

The DataBag abstraction serves as a coarse-grained contract for data-parallel computation. The deep embedding in Scala gives Emma the ability to manipulate the entire data analysis program at compile time, which has several key benefits: First, Emma reuses Scala's for-comprehension syntax for declarative, SQL-like dataflow definitions. Second, Emma is able to decompose the data analysis program into a sequential driver part and and multiple parallel dataflow fragments. These fragments can then be optimized jointly and then translated and executed on parallel dataflow engines like Apache Spark or Flink. With this, Emma exposes a high-level collection processing API to the user who wants to parallelize a data analysis program, while hiding the notions of parallelism associated with the underlying dataflow engines.

As is depicted in Figure 1, Emma first lifts the Abstract Syntax Tree (AST) of the data analysis program to be parallelized to a suitable intermediate representation (IR) and performs logical as well as physical optimizations.

This rewritten intermediate representation is then lowered and compiled as a driver with abstract dataflow expressions. At runtime, these dataflow expressions are translated *Just in Time (JIT)* and evaluated on the target dataflow engines Apache Flink or Spark. These algebraic rewrites and physical optimizations lead to competitive performance compared to hand-tuned low-level code [2] while hiding the notions of parallelism associated with the underlying dataflow engines from the user. For example, Emma will also transparently place primitives like broadcast and cache that influence physical execution aspects of the data flow that would have to be specified by the programmer themselves.

While Emma relieves the programmer from having to understand systems internals of the chosen execution engine the `DataBag` abstraction may still make it unnecessarily complicated to encode complex data analysis and machine learning algorithms. To ease the encoding and improve the readability and interpretability of these algorithm, BBDC researchers developed *LARA* [11] a deeply embedded language in Scala which adds two abstract data types for linear algebra, called `Matrix` and `Vector`, on top of the `DataBag` type provided in Emma. Thus, it unifies aspects of relational algebra and linear algebra. LARA provides explicit operations to convert a `Databag` to a `Matrix` and vice versa. The `Matrix` abstraction provides all common operations on matrices and is strongly influenced by R's matrix package. Similar to Emma, data analysis programs written in Lara are compiled to an intermediate representation and then optimized, this time enabling optimizations across linear and relational algebra.

One example of such an optimization considers the partitioning of matrices in distributed data flow engines like Spark and Flink, which traditionally implement relational operators on row-partitioned datasets. Linear algebra operators tend to use block-partitioned matrices for efficiency reasons. A holistic optimization approach over pipelines combining both kinds of operators can actually avoid expensive re-partitioning steps by fusing relational and linear algebra operations into a specialized physical operator called BlockJoin [12], a distributed join algorithm, which directly produces block-partitioned results.

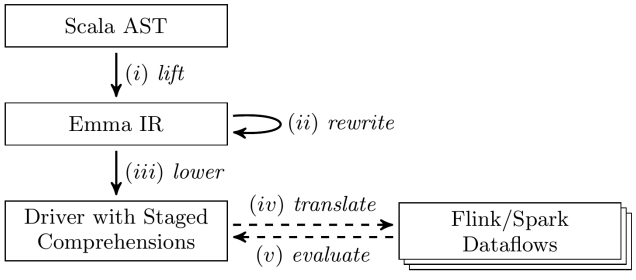

Fig. 1. The Emma compilation pipeline.

3 Benchmarking Distributed Data Processing Systems: PEEL

Another very important consideration in the context of distributed data processing systems but also scalable machine learning applications is the reproducibility of experiments and evaluations. The experiments published in scientific publications associated with distributed data processing systems such as Spark [20] or Stratosphere [1] are only of limited use when it comes to adequately assessing and comparing the performance of different approaches and paradigms. Different evaluation workloads and implementations, usage of potentially system-specific libraries, different or custom pre-processed data sets and differing hardware configurations make it hard if not impossible to leverage the published experiments for such a comparison.

In order to foster transparent, transferable and, thus, repeatable and reproducible experiments and benchmarks of distributed data processing systems, the BBDC researchers developed a framework called *PEEL*[6] [5] de-

6 http://peel-framework.org/

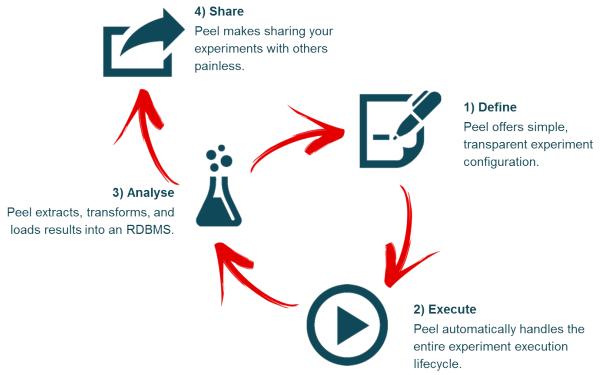

Fig. 2. The Peel process

signed to define, execute, analyze, and share experiments. PEEL introduces a unified and transparent process of specifying experiments illustrated in Figure 2, including the actual experiment workload code, all relevant system configuration parameters for all systems involved (e.g. distributed file system, distributed data processing system, data generators) and the experiment description including all parameters to be varied as part of the experiment. PEEL automatically orchestrates all specified benchmark experiments and handles setup, configuration, deployment, tear-down and cleanup of all required systems and automatically collects all relevant logs from the remote machines involved.

Next to specifying systems experiments to evaluate distributed data processing systems, PEEL is also a useful tool to organize the scalable machine learning algorithms that have been developed in the context of the BBDC. We have successfully used PEEL to develop and conduct comprehensive benchmark experiments to evaluate distributed data flow systems for scalable machine learning workloads. Results indicate that while they do exhibit the desired scaling behavior with respect to the number of compute nodes and data set sizes, distributed data flow systems introduce substantial runtime overhead compared to efficient single machine implementations [6] and tend to struggle with training high-dimensional models [7].

4 Scalable Machine Learning

Besides the research work focusing on scalable data management, the Berlin Big Data Center also produced many interesting results in the fields of scalable machine learning [9, 18] and the associated use cases in the fields of video mining, text analytics, information-based medicine, and material science.

For example, in the application area of material science, BBDC researchers developed a data-analysis tool for the recognition of the similarity among crystal structures and for the prediction of the difference in formation energy among them based on the Novel Materials Discovery (NOMAD[7]) Archive. Nomad is a large open-access repository for computational materials science data that contains several millions of crystal configurations. The similarity-recognition algorithm, based on descriptors that encode the proper symmetries of a well-behaved physical representation and makes use of linear and non-linear low-dimensional embedding methods, produces a 2-dimensional map that assigns to separate regions perfect and distorted configurations, for given pairs of crystal structures. The algorithm predicting the difference in formation energies selects the model out of thousands of candidates, by means of a compressed-sensing based method [10].

In the context of video mining, BBDC researchers worked on human action recognition algorithms that work in the compressed video domain [19]. These algorithms are extremely efficient, because they only require a partial decoding of the video bit stream. An important part of this work is to make the decisions of these approaches interpretable to humans. To enable this, BBDC researchers developed Layer-Wise Relevance Propagation (LRP)[4, 14], an approach to understand the contribution of a single pixel of an image to the prediction made by a classifier in an image classification or human action recognition task. This method has also been applied to understand and interpret action recognition algorithms and apply it to a state-of-the-art compressed domain method based on Fisher vector encoding and SVM classification [13], visualizing what exactly makes the algorithm decide for a particular action class.

5 Vision and Challenges of BBDC Phase II

The Berlin Big Data Center has recently been extended for a second phase lasting another three years. In this section we will sketch the challenges to be addressed during this phase. In general the research goals are grouped into four core areas:

7 https://www.nomad-coe.eu/

– declarative machine learning in selected applications;
– machine learning on heterogeneous data and data streams;
– scalable processing of heterogeneous, geographically distributed data streams; and
– near real-time processing of millions of data sources.

The selected application domains include the analysis of distributed bio-medical mass data, the analysis of heterogeneous data in cancer research, learning on compressed stream data and near real-time language processing. In the context of machine learning on heterogeneous data and data streams, work will focus on reproducible [16] as well as secure machine learning, data security and privacy as well as the automatic integration of heterogeneous data. The work on scalable processing of heterogeneous, geographically distributed data streams will address geo-distributed, declarative data analysis, consistent state management resource management optimized processing of semantic graphs, error analysis of data analysis programs as well as system integration and performance analysis [17]. Finally the area of near real-time processing of millions of data sources will include data stream analysis through Software-Defined Networking (SDN), data stream processing on modern hardware as well as dealing with millions of sensors

6 Conclusion

In this article, we introduced several key results of the Berlin Big Data Center including the domain-specific language Emma, which is deeply embedded in Scala and enables parallel collection processing through for comprehensions and LARA which offers abstract data types for both relational and linear algebra processing. Next to these scalable data management approaches, we also presented work on interpretability of machine learning models and sketched the key challenges to be addressed during the second phase of the Berlin Big Data Center.

Acknowledgment: This work has been supported through grants by the German Ministry for Education and Research as Berlin Big Data Center BBDC (funding mark 01IS14013A).

References

[1] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *The VLDB Journal*, 23(6), Dec. 2014.

[2] A. Alexandrov, A. Kunft, A. Katsifodimos, F. Schüler, L. Thamsen, O. Kao, T. Herb, and V. Markl. Implicit parallelism through deep language embedding. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 47–61, 2015.

[3] A. Alexandrov, A. Salzmann, G. Krastev, A. Katsifodimos, and V. Markl. Emma in action: Declarative dataflows for scalable data analysis. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 2073–2076, New York, NY, USA, 2016. ACM.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015.

[5] C. Boden, A. Alexandrov, A. Kunft, T. Rabl, and V. Markl. Peel: A framework for benchmarking distributed systems and algorithms. In *Proceedings of the Ninth TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC 2017) at VLDB 2017*, 2017.

[6] C. Boden, T. Rabl, and V. Markl. Distributed machine learning - but at what cost? In *ML Systems Workshop @ NIPS 2017*, MLSystems'17, 2017.

[7] C. Boden, A. Spina, T. Rabl, and V. Markl. Benchmarking data flow systems for scalable machine learning. In *Proceedings of the 4th Algorithms and Systems on MapReduce and Beyond*, BeyondMR'17, pages 5:1–5:10, New York, NY, USA, 2017. ACM.

[8] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.*, 38(4):28–38, 2015.

[9] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5), 2017.

[10] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler. Big data of materials science: Critical role of the descriptor. *Phys. Rev. Lett.*, 114:105503, Mar 2015.

[11] A. Kunft, A. Alexandrov, A. Katsifodimos, and V. Markl. Bridging the gap: Towards optimization across linear and relational algebra. In *Proceedings of the 3rd ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond*, BeyondMR '16, pages 1:1–1:4, New York, NY, USA, 2016. ACM.

[12] A. Kunft, A. Katsifodimos, S. Schelter, T. Rabl, and V. Markl. Blockjoin: Efficient matrix partitioning through joins. *Proc. VLDB Endow.*, 10(13):2061–2072, Sept. 2017.

[13] S. Lapuschkin, A. Binder, G. Montavon, K. R. Müller, and W. Samek. Analyzing classifiers: Fisher vectors and deep neural networks. In *2016 IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 2912–2920, June 2016.

[14] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[15] T. Rabl, J. Traub, A. Katsifodimos, and V. Markl. Apache flink in current research. *it - Information Technology*, 58(4):157–165, 2016.

[16] S. Schelter, J.-H. Boese, J. Kirschnick, T. Klein, and S. Seufert. Automatically Tracking Metadata and Provenance of Machine Learning Experiments. *Machine Learning Systems workshop at NIPS*, 2017.

[17] S. Schelter, J. Soto, V. Markl, D. Burdick, B. Reinwald, and A. Evfimievski. Efficient sample generation for scalable meta learning. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1191–1202, 2015.

[18] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017.

[19] V. Srinivasan, S. Lapuschkin, C. Hellge, K. R. Müller, and W. Samek. Interpretable human action recognition in compressed domain. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1692–1696, March 2017.

[20] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. NSDI'12, 2012.

List of contributors

Christoph Boden
TU Berlin, DIMA, Einsteinufer 17, 10587 Berlin
christoph.boden@tu-berlin.de

Christoph Boden is a research associate and PhD student at the Database Systems and Information Management group at TU Berlin and at the German Research Center for Artificial Intelligence (DFKI). He is part of the Management of the Berlin Big Data Center (BBDC). In his research he focuses on benchmarking data processing systems for scalable machine learning workloads.

Tilmann Rabl
TU Berlin, DIMA, Einsteinufer 17, 10587 Berlin
rabl@tu-berlin.de

Tilmann Rabl is a professor at the Database Systems and Information Management (DIMA) group at TU Berlin and at the German Research Center for Artificial Intelligence (DFKI). At DIMA he is research director and coordinator of the Berlin Big Data Center (BBDC). Tilmann Rabl is also cofounder of the startup bankmark.

Volker Markl
TU Berlin, DIMA, Einsteinufer 17, 10587 Berlin
prof@dima.tu-berlin.de

Volker Markl is a Full Professor and Chair of the DIMA Group at TU Berlin and an Adjunct Full Professor at the University of Toronto. He is Director of the Intelligent Analytics for Massive Data Research Group at DFKI and Director of the Berlin Big Data Center.