A Survey of Big Data, High Performance Computing, and Machine Learning Benchmarks

Nina Ihde[1], Paula Marten[1], Ahmed Eleliemy[2], Gabrielle Poerwawinata[2], Pedro Silva[1], Ilin Tolovski[1], Florina M. Ciorba[2], and Tilmann Rabl[1]

[1]Hasso Platner Institut, Potsdam, Germany
[2]University of Basel, Switzerland
{pedro.silva, ilin.tolovski, tilmann.rabl}@hpi.de,
{nina.ihde, paula.marten}@student.hpi.de,
{ahmed.eleliemy, gabrielle.poerwawinata, florina.ciorba}@unibas.ch

Abstract. In recent years, there has been a convergence of Big Data (BD), High Performance Computing (HPC), and Machine Learning (ML) systems. This convergence is due to the increasing complexity of long data analysis pipelines on separated software stacks. With the increasing complexity of data analytics pipelines comes a need to evaluate their systems, in order to make informed decisions about technology selection, sizing and scoping of hardware. While there are many benchmarks for each of these domains, there is no convergence of these efforts. As a first step, it is also necessary to understand how the individual benchmark domains relate.
In this work, we analyze some of the most expressive and recent benchmarks of BD, HPC, and ML systems. We propose a taxonomy of those systems based on individual dimensions such as accuracy metrics and common dimensions such as workload type. Moreover, we aim at enabling the usage of our taxonomy in identifying adapted benchmarks for their BD, HPC, and ML systems. Finally, we identify challenges and research directions related to the future of converged BD, HPC, and ML system benchmarking.

Keywords: Benchmarking · Big Data · HPC · Machine Learning.

1 Introduction

A benchmark refers to a process to obtain quantitative measures that enable meaningful comparison across multiple systems [19]. Such quantitative measures are essential to explore and assess the potential benefits and drawbacks of emerging software and hardware architectures. Benchmarks range from simple computational kernels, mini-apps, and proxy-apps to full applications, which are used to stress one or more components of the system under test (SUT).

In this work, we turn our attention to big data (BD), high performance computing (HPC) and machine learning (ML) systems which have been fueled by modern applications that rely on complex data analytics pipelines. Besides "classic" applications such as stream processing for BD, complex simulations for HPC and neural network training for ML, we also observe a growing number of applications that belong to two or more of those domains, such as digital twins [33] or earning simulation engines in the context of Industry 4.0 [60]. Those applications require *hybrid-systems* that lie at the *convergence* between BD, HPC, and ML [45,59,26].

The state-of-the-art benchmarks that either target BD, HPC, or ML systems are rich. However, the appearance of those hybrid systems raises questions about the need for developing new benchmarks capable of evaluating them.

Our objective in this work is to review the literature on BD, HPC, ML, and hybrid benchmarks and investigate their capabilities. Hybrid systems include specific hardware and software components optimized for BD, HPC, and ML workloads. Consequently, hybrid benchmarks refer to benchmarks that can assess the performance of hybrid systems. We propose a classification of modern and widely used BD, HPC, and ML benchmarks using a *feature space* composed of *purpose*, *stage level*, *metrics*, and *convergence* which allow us to perform a unified analysis of BD, HPC, ML, and hybrid benchmarks. The features space is complemented by a high level architecture of modern *data analysis pipelines* that helps visualizing the capabilities of the evaluated benchmarks and drawing insights on improvement directions.

The rest of this article is organized as follows. In Section 2, we discuss some of the most representative and modern BD, HPC and ML benchmark systems. In Section 3, we present our classification methodology and use it to classify the work discussed in Section 2. In Section 4, we highlight certain research efforts that are closely related to the current work. In Section 5, we discuss our insights, present our vision on convergence, and conclude this work.

2 Background

Benchmarking refers to the process of obtaining quantitative measures that enable performance comparison across a set of target systems or components [19]. Benchmarks are critical to explore the potential benefits and drawbacks of emerging software and hardware architectures, and the vast amount of existing benchmarks/benchmark suites reflects their importance to the scientific community. HiBench [41], BigBench [36], BigDataBench [34], YCSB [27], LDBC Graphalytics [42] are examples of BD benchmarks. NPB [15], SPEC [10,8,10], HPCC [47], UEABS [11] CORAL-2 [2], and HPCG [29] are examples of HPC benchmarks. DeepBench [53], MLPerf [49], Fathom [12], LEAF [24], and CleanML [46] are examples of ML benchmarks. The point of these lists is not to be exhaustive but rather to show the exuberance of existing benchmarks, which yields several challenges regarding selecting a specific benchmark. In the following, we survey the most common and well-known benchmarks for BD, HPC, and ML systems.

2.1 Big Data Benchmarking

The main characteristics of BD systems are summarized in the "Vs" of BD, *volume*, *variety*, and *velocity*, these must be captured by workloads and metrics of BD benchmarks. Typically, BD workloads are characterized by processing large amounts of data that may be encoded in different types, such as text, semi-structured data, structured data, or binary, and may be delivered in different speeds, such as statically for batching, or streamed. The metrics are commonly related to the amount of data processed, the time taken, and the resources used for processing.

In the next paragraphs, we describe in more detail some of the most important BD benchmarks in the literature, their workloads and metrics.

2.1.1 HiBench HiBench [41,40,39] is an open-source BD benchmark proposed by Intel. It focuses on offering workloads based on real-world use cases that could be representative of situations found when processing large amounts of data. Therefore, HiBench proposes 29 different workloads in 6 different categories.

Initially designed to be a benchmark for Apache Hadoop[1] in 2010, HiBench would originally support four MapReduce related workloads categories. Nevertheless, HiBench has been continuously updated since then and currently offers 6 workload categories: (i) external sorting and file system micro-benchmarks, (ii) machine learning (e.g., K-means and Bayesian Classification), (iii) web search (e.g., PageRank [22]), (iv) OLAP, (v) graph processing, and (vi) stream processing. The metrics considered by HiBench are resource consumption (e.g. CPU, I/O, Memory), job running time, throughput, HDFS bandwidth, and data access patterns (e.g., ratio amount of input data per amount of output data).

Finally, HiBench also has pre-configurations [41] for running sets of workloads on updated versions of Apache Hadoop and Apache Spark[2], and on old versions of the stream processing systems Apache Flink[3], Apache Storm[4], and Apache Kafka[5].

2.1.2 BigBench BigBench [36] is an end-to-end BD benchmark and the basis of TPCx-BB [18]. It is based on TPC-DS [56], a decision support benchmark, and proposes a rich data model and a generator covering the key aspects of BD systems *volume*, *variety* and *velocity*. That is accomplished through the generation of large amounts of data that may be structured, semi-structured or unstructured at high frequencies by an extension of the Parallel Data Generation Framework [57] (PDGF) data generator.

BigBench's workload is inspired by a BD retail business analytics by McKinsey and encompasses the *collect*, *store*, and *analysis* steps of a BD system

[1] https://hadoop.apache.org/
[2] https://spark.apache.org/
[3] https://flink.apache.org/
[4] https://storm.apache.org/
[5] https://kafka.apache.org/

life-cycle. It is composed of 30 queries designed to cover different (i) *data inputs*, (ii) *processing types*, and (iii) *analytic techniques*. In terms of data input, it can generate and load structured data (database tables), semi-structured data (logs), and unstructured data (natural language reviews). The supported processing types relate to the type of paradigm that is most adapted to answering a query, and can be procedural (e.g., Map Reduce programs) or declarative (e.g., SQL). Finally, the analytic techniques define the approach for solving a query, which are statistical analysis (e.g., linear regression), data mining (e.g., clustering) or simple ad-hoc queries (e.g., simple SQL queries).

The metrics used by BigBench are based on those used by TPC-DS, and are, in summary, the partial execution times of the different steps of the benchmark. It also proposes a final metric defined as a geometric mean [18] of the partial execution times.

2.1.3 BigDataBench

BigDataBench[6] [34] is a BD and Artificial Intelligence (AI) benchmark suite mainly provided by the Institute of Computing Technology (Chinese Academy of Sciences) and the non-profit organization BenchCouncil. BigDataBench is open-source and has been actively developed since 2013 [35,64,38,61]. Most recently, version 5.0[7] was released, which covers five application domains (search engine, social networks, electronic commerce, multimedia processing, bioinformatics) and defines a workload as a pipeline of one or more so-called "data motifs". This refers to classes of units of computation that typically consume the majority of the runtime of workloads. Especially when these data motifs are used together, it should be possible to address a wide range of BD and AI workloads. So instead of applying a separate benchmark for each workload, the authors of BigDataBench suggest using data motif-based workloads, which is why they have elaborated eight data motifs, namely Matrix, Sampling, Logic, Transform, Set, Graph, Sort and Statistic computation.

BigDataBench contains a BD Generator Suite (BDGS) [51] that can generate synthetic data based on scaled real data. Both the synthetic and the additional 13 real-world data sets can be structured, semi-structured or non-structured. Moreover, they are extracted from text, graph, table, or image data to model the impact of different data types and inputs on the workload's behaviour and runtime.

Furthermore, BigDataBench offers 44 BD and AI benchmarks with respective implementations for seven workload types (online services, offline analytics, graph analytics, AI, data warehouse, NoSQL, and streaming). Three types of benchmarks are provided: (i) *micro benchmarks*, (ii) *component benchmarks*, and (iii) *end-to-end application benchmarks*. A single data motif represents a micro benchmark, for instance sort (offline analytics), filter (data warehouse) or connected component (graph analytics). If several data motifs are put together like

[6] https://www.benchcouncil.org/BigDataBench/
[7] https://www.benchcouncil.org/BigDataBench/files/BigDataBench5.0-User-Manual.pdf

for clustering or classification benchmarks, this is called a component benchmark. Several component benchmarks form an end-to-end application benchmark. Finally, BigDataBench offers a benchmark model for the examination of the hardware, software, and algorithms. Each of the models considers at least as metrics the wall clock time and energy efficiency to execute a benchmark.

2.1.4 YCSB

The Yahoo! Cloud Serving Benchmark (YCSB)[8] [27,17] is an open-source benchmark suite for cloud data serving systems started in 2010. The main components of this benchmark are the *YCSB client* for the generation of workloads and the core package of workloads. When the YCSB client decides for a operation to perform (insert, update, read, scan, or delete) or how many and which records to scan, these decisions are based on random distributions (e.g. uniform or multinomial). Accordingly, each of the 6 available workloads in the core package is defined by one or several distribution/s which is/are combined with a set of operations to perform and records to read or write. The offered workloads by the YCSB are (i) update-heavy, (ii) read-heavy, (ii) read-only, (iv) read latest, (v) read-modify-write, and (vi) short range scan. One can observe that the core package consists of related workloads that stress a wider range of the performance space than a single workload which can examine a system at one specific point in the performance space. In addition, the framework is easy expandable with new workloads that enable the benchmarking of new systems.

The YCSB mainly evaluates the performance and the scalability of the cloud data serving the system under test. For measuring the performance, the latency of requests is monitored while the throughput (i.e. the load of the database) is increased. The scalability is benchmarked with two different metrics. The first metric *scaleup* looks at how the system behaves as the amount of servers is increased. This is implemented by loading the servers with data and starting the workload. The data is then removed to add more servers, which then run the workload again with more data. The second metric *elastic speedup* does basically the same, but here the servers are added while the workload is still running.

2.1.5 LDBC Graphalytics

Linked Data Benchmark Council (LDBC) Graphalytics[9] [42] is a benchmark suite for graph analysis platforms which has been under constant development since 2014 [37,25,55]. Its construction is mainly based on the approach that a benchmark should have diversity in terms of (i) *algorithms*, (ii) *data sets*, and (iii) *metrics*. Therefore it currently consists of six algorithms, namely BFS, PageRank [23], weakly connected components, community detection using label propagation, local clustering coefficient, and single-source shortest path. The algorithms were selected in collaboration with the LDBC Technical User Community (TUC) and using surveys to ensure that they are the most relevant and cover a wide variety of scenarios.

Second, Graphalytics consists of synthetic and real data sets categorized into “T-shirt size” classes of different magnitudes (e.g., graphs in scale from 7.5 to 8

[8] https://github.com/brianfrankcooper/YCSB
[9] https://graphalytics.org/

belong to the class "S"). The criteria for the data set choice are size, domain and characteristics. In order to make the creation of synthetic data sets possible, the LDBC Social Network Benchmark data generator (Datagen)[10] was extended.

Third, the Graphalytics benchmark suite offers a test harness which allows to run a large set of diverse experiments. Thus, scalability and performance are evaluated using deep metrics. For instance, scalability is measured in terms of whether the data set size grows when using more resources (strong vs. weak scaling). In addition, robustness of the system under test can be specified with performance variability, crash points and service-level agreement (SLA) compliance. The SLA is fulfilled if the algorithm execution for a given data set requires a maximum of one hour.

Complementary to the algorithms, data sets, and metrics that Graphalytics provides, it offers reference outputs for a standardized comparison of different platforms and reference implementations for graph analysis platforms from the community (e.g., Giraph[11]) and from the industry (e.g., PGX [6]). Industrial platforms were benchmarked by the vendors themselves. In order to constantly meet the performance requirements and general developments of graph analysis platforms, a new version of Graphalytics is published every two years, which mainly adapts the workload (i.e., the chosen data sets and algorithms).

2.2 High Performance Computing Benchmarking

The goal of HPC benchmarking is to evaluate the HPC system performance and application characteristics. HPC workloads stress different system components such as CPU, memory, I/O, network, file-system, etc. The response of these system parts is measured with different performance metrics, such as speedup, throughput, communication speed, access speed, and others. The workloads typically originate from computational science domains, such as physics, chemistry, material science, etc. The evaluation of existing systems also creates a yardstick for the design of future HPC systems. We summarize classical HPC benchmarks in the following subsections.

2.2.1 NPB The Numerical Aerodynamic Simulation Parallel Benchmarks (NPB)[12] by NASA Ames Research Center, is a large effort to advance the state of computational aerodynamics [14]. NPB 1.0 was introduced in 1992; later another version (NPB 2.3) was introduced in 1997 [66] and included MPI implementations. NPB includes five kernels: (i) Embarrassingly Parallel (EP) that assesses the floating point computing rate. (ii) MultiGrid (MG), (iii) Conjugate Gradient (CG), (iiii) Discrete 3D Fast Fourier Transform (FT), and Large Integer Sort. (IS) which all evaluate the integer computation rate. NPB contains three pseudo applications: (i) Lower-upper Gauss-Seidel (LU) to evaluate fine-grained MPI communication [15], (ii) Block Tri-diagonal (BT) and (iii) Scalar

[10] https://www.ldbcouncil.org/ldbc_snb_docs/ldbc-snb-specification.pdf
[11] https://giraph.apache.org/
[12] https://www.nas.nasa.gov/software/npb.html

Penta-diagonal (SP) to measure coarse-grained MPI communication [15]. NPB defines performance results in the number of floating-point operations per second (FLOP/s). NPB codes are written in Fortran-77 or C.

While NPB benchmarks exhibit fine-grained exploitable parallelism, many scientific problems require several levels of parallelism. Therefore, the NPB Multi-zone was introduced [65]. Aside from improvements were made to parallel systems including the scalability and performance, the limitations on I/O performance were evident due to access to data files. Thus, BTIO was invented based on BT benchmark using MPI-IO and were used to test the speed of parallel I/O [66]. Another benchmark called Arithmetic Data Cube (ADC) extends typical data mining operations related to Data Cube Operator in OLAP tool into a grid environment. ADC measures data movement across computational grid and across memory hierarchy of individual grid machines [32].

2.2.2 SPEC Standard Performance Evaluation Corporation (SPEC)[13] was founded in 1988 and is a non-profit consortium that has 22 major computer vendors whose common goals are to provide the industry with performance measurement tools and educate consumers about the performance of vendors' products [48]. The development of the benchmark suites since 1994 includes obtaining candidate benchmark codes, putting these codes into the SPEC harness, testing and improving the codes' portability across many operating systems, compilers, interconnects, runtime libraries, and for correctness and scalability [52]. We consider SPEC CPU 2017 and SPEC HPC: OpenMP, MPI, OpenACC, OpenCL benchmarks are the most useful to HPC. SPEC CPU 2017 and SPEC HPC benchmarks code were written in C, C++, or Fortran.

SPEC CPU 2017 focuses on the performance of compute-intensive applications on processor, memory systems, and compilers. SPEC CPU 2017 comprises 43 benchmarks, organized into four suites: SPECspeed Integer and SPECspeed Floating Point suites both employ time (in seconds) as a metric to compute single tasks; SPECrate Integer and SPECrate Floating Point both measure the throughput or work per unit of time metric (jobs per hour) [10]. SPEC High Performance Computing consists of SPEC ACCEL measures parallel compute performance including the hardware accelerator, the host CPU, the memory transfer between host and accelerators, and the compilers [7]; SPEC MPI 2007 compares measurement of MPI-parallel, floating-point, compute-intensive performance, across the widest range of cluster and SMP hardware [9]; SPEC OMP 2012 measures the performance of applications based on the OpenMP 3.1 standard for shared-memory parallel processing [8].

2.2.3 HPCC The HPC Challenge (HPCC)[14] benchmark suite was developed for the DARPA's HPCS (High Productivity Systems) Program to provide a set of standardized hardware probes based on commonly occurring computational

[13] https://www.spec.org/benchmarks.html

[14] https://icl.utk.edu/hpcc/

software kernel [43]. The suite is designed to augment the Top500 list, providing benchmarks that bound the performance of many real applications as a function of memory access. HPCC's first version was released in 2003. The HPCS program performance targets will flatten the memory hierarchy, improve real application performance, and decrease development time. The suite is composed of several computational kernels such as STREAM (PB/s), HPL (Pflop/s), matrix multiply- DGEMM, parallel matrix transpose - PTRANS, and FFT (Pflop/s), that attempt to span high and low spatial and temporal locality space [47].

2.2.4 UEABS The Unified European Application Benchmark Suite (UEABS)[15] was released during the Partnership for Advanced Computing in Europe Second Implementation Phase (PRACE-2IP) project in 2010 [11], which consist of ALYA[15] that solves computational mechanics models, Code_Saturne[15] that is a multi-purpose CFD software, CP2K[15] which performs atomistic simulations, GADGET[15] which simulates cosmological N-body/SPH, GPAW[15] and Quantum Espresso[15] which calculates electronic structure, GROMACS[15] and NAMD[15] which simulates molecular dynamics, PFARM[15] which solves many-electron equation program, NEMO[15] which models ocean and climate sciences, QCD[15] which performs quantum chromodynamics, SHOC[15] that tests the performance and stability of systems, SPECFEM3D[15] which simulates seismic wave, and DeepGalaxy[15] using Tensorflow which performs neural-networks optimization in machine learning [11].

Each benchmark has their own test case/s and were experimented over PRACE Tier-0 and Tier-1 Systems, PRACE PCP, DEEP-ER, and Mont-Blanc 3 Prototype machines [63]. The metrics such as time(s), speedup, parallel efficiency, energy consumption (kJ), and performance in nano seconds elapsed per day (ns/day) are the output from this suite [63]. The benchmarks programming languages are as follows: ALYA, Code_Saturne, QuantumEspresso, SPECFEM3D, NEMO were written in Fortran 90/95; PFARM in Fortran 2003; CP2K in Fortran 2008; Code_Saturne, GADGET, GPAW, and GROMACS were written in C; NAMD and NEMO in C++; GPAW and DeepGalaxy were written in Python [11]. Parallelization using MPI is used in CP2KL, GADGET, GPAW, GROMACS, NAMD, PFARM, QCD, and SPECFEM3D. OpenMP and MPI are used in ALYA, Code_Saturne, PFARM, and QuantumEspresso; multi-threading is used in CP2K; GPU support applies to CP2K, NAMD, PFARM, GROMACS v4.6, QCD, SHOC, and DeepGalaxy; and TPUs support is provided in DeepGalaxy [11].

2.2.5 CORAL-2 The collaboration of Oak Ridge, Argonne, and Lawrence Livermore National Laboratory (CORAL) began in late 2012 intending to deliver three systems that would each improve delivered performance as compared to the existing 20 PetaFlop Department of Energy (DOE) systems - Titan at Oak Ridge National Laboratory (ORNL) and Sequoia at Lawrence Livermore Na-

[15] https://repository.prace-ri.eu/git/UEABS/ueabs/

tional Laboratory (LLNL) [62]. The CORAL-2 benchmarks[16] are distinguished into four categories: scalable science, throughput, data science and deep learning, and skeleton benchmarks.

The scalable science benchmarks are expected to run at full scale of the CORAL systems [2], and include CORAL-2 HACC, Nekbone, LAMMPS (atom-timesteps/s), and QMCPACK (samples/s). The throughput benchmarks (metric: total number of unknowns (across all MPI tasks)/(sec/iter)) represent large ensemble runs, and applications include CORAL-2 AMG, Kripke, Quicksilver, and PENNANT.

The data science and deep learning benchmarks contain machine learning (TB/s) with K-Means, PCA, and SVM; and deep learning algorithms (batch/s) with Convolutional Neural Networks (CNN) and Recurrent Neural Network (RNN). Lastly, skeleton benchmarks investigate various platform characteristics including performance on network, threading overheads, I/O, memory, system software and programming models [2].

2.2.6 HPCG The High-Performance Conjugate Gradient (HPCG) benchmark[17] performs a fixed number of multigrid preconditioned (using a symmetric Gauss-Seidel smoother) conjugate gradient (PCG) iterations using double precision (64-bit) floating point values [3]. High Performance Linpack (HPL)[18] solves a (random) dense linear system in 64-bit arithmetic on distributed-memory computers [4] and measures the sustained floating-point rate (GFLOP/s). HPCG has been released as a complement to the FLOPs-bound HPL [58].

In 2019, the High Performance Linpack-Artificial Intelligence (HPL-AI)[19] benchmark seeks to highlight the emerging convergence of HPC and AI workloads [5]. The development of HPL-AI is to experiment HPC simulations on new accelerator hardware [44]. HPL-AI strives to unite modern algorithms and contemporary hardware while connecting its solver formulation to the decades-old HPL framework [5]. Some of the machines on the TOP500 have been benchmarking with HPL-AI, such as RIKEN's Fugaku supercomputer which HPL-AI achieved 2 EFLOP/s, while HPL achieved 0.442 EFLOP/s [5].

2.3 Machine Learning Benchmarking

ML systems are characterized by the usage of complex ML models for specific ML inference tasks, such as classification or regression. Those models are often *trained* using large amounts of data which commonly have to be pre-processed first. Therefore, we can delineate *data preparation*, *training*, and *inference* as the most common stages of ML systems and the main targets of modern ML benchmarks. The workload of each of those stages vary, as well as their performance

[16] https://asc.llnl.gov/coral-2-benchmarks
[17] https://www.hpcg-benchmark.org/index.html
[18] https://www.netlib.org/benchmark/hpl/
[19] https://icl.bitbucket.io/hpl-ai/

metrics. Data preparation is usually a processing intensive step involving transforming data into a format that can be exploited by other stages, hence, resource consumption and execution time are common performance metrics. Training is also a very processing intensive stage, however, the objective is often to achieve a certain model accuracy, thus metrics related to time-to-accuracy are an addition to resource consumption and execution time. In the inference stage, the accuracy of the model with unseen data and its latency to calculate solutions are the main performance indicators.

In the next paragraphs we analyze some of the main ML benchmarks in the state-of-the-art and discuss in more details stages, workloads and metrics considered by them.

2.3.1 DeepBench [53] is a ML benchmark designed by Baidu Research with an open source implementation[20]. It measures the performance of hardware systems on basic operations necessary for deep neural network training and inference. It was first released in 2016 and extended a year later to account for broader operation types and to include inference [54].

DeepBench benchmarks four different operations: Matrix multiplications, convolutions, recurrent layers and all-reduce. It allows for the transposition of matrices and supports vanilla, long short term memory or gated recurrent unit architecture for convolutions [53] DeepBench provides a selection of numerical data that the operations are run on, all varying in size[21]. It allows the evaluation of these operations on dense and sparse matrices and vectors in order to also evaluate systems with regards to sparse neural network training and inference [54]. The minimum precision for all accumulating operations is 32 bits, for multiplication in training and inference operations the precision has to be at least 16 and 8 bits respectively. The performance for each operation is measured in TeraFLOPS and milliseconds. For sparse operations the speedup in comparison to the same operation on dense matrices is also provided.

2.3.2 MLPerf [49] is a benchmark suite that can measure a system's performance for training and inference in ML pipelines. The suite offers customized benchmarks for both cases under different hardware configurations, suitable for proof-testing embedded and mobile devices, single nodes, and distributed data centers.

The MLPerf suite covers five different ML area: image classification, object detection, translation, reinforcement learning and recommendation. One data set is provided for each task besides the reinforcement learning task, where the data is generated in training. The data set used for the recommendation task is the only synthetic set. Systems can be compared in two versions of MLPerf: In the closed division the benchmark specifies which model to use on which task, only for object detection and translation it is possible to choose between two models,

[20] `https://github.com/baidu-research/DeepBench`
[21] `https://github.com/baidu-research/DeepBench`

which are fit for different applications and represent a broader collection of ML models. The open division of the benchmark allows free model choice [50].

MLPerf training uses time-to-train to a defined accuracy as a performance metric [50]. This end-to-end metric provides a combined measurement of the speed and accuracy of a system. The goal-accuracy chosen for each model is slightly beneath the state-of-the-art to, show adverse effects of speed optimizations on the quality of a model, and reduce run-to-run variations while still being able to achieve the goal-accuracy in each run. MLPerf perform many runs of the tasks in order to reduce the effects of run-to-run variance. However, certain parts of the training process are excluded from the timing: System initialization, model creation and initialization taking up to 20 minutes and data reformatting, including organizing the data in databases or changing file formats. In the inference benchmarks, MLPerf measures the throughput of queries and samples per second, with respect to latency constraints. Multiple latency constraints can be imposed when evaluating distributed or multistream systems. By implementing the latency constraints in their inference benchmarks, MLPerf can proof-test systems in varying conditions adapted for specific use cases.

All implementations benchmarked must be mathematically equivalent to the reference implementation, which are provided either in PyTorch or Tensor Flow[22]. Adjustment of the hyper-parameters are only allowed inside of a defined frame. These rules should ensure that the actual system performance is measured and not any user optimization or hyper-parameter exploration abilities.

2.3.3 Fathom [12] is a collection of reference workloads of deep learning models. It includes open source implementations of seven ML models, all representatives of the state-of-the-art in deep learning, offering a diverse set of characteristics and influential to deep learning research. All open source implementations of the workloads [23] are provided in TensorFlow and were either adapted, translated from a different language or re-implemented to fit their description in the original paper. Pre- or post-processing steps and extra logging was removed from the original implementations. The implementation also provides a standard model interface and is compatible with Fathom's profiling tools. The workloads are either run on the data used in the original implementation of the model or on an open source alternative which is widely used for the type of model. Fathom includes recurrent neural networks, memory networks, auto-encoders, residual networks, convolutional networks and deep reinforcement learning models. To asses the performance of the hardware and system running the model the execution time of primitive operations during training and inference is measured. These operations, including matrix multiplies and tensor exponentiations, are the smallest schedulable unit in TensorFlow and make up up to 99% of the runtime. Measuring their execution time can not only offer insight into the time spent on a certain operation, but also show similarities between workloads, effects of parallelism and the correlations and ratios between different types of

[22] https://github.com/mlcommons/training
[23] https://github.com/rdadolf/fathom

operations. Using this data the qualities and trade offs of hardware and systems developed for ML can be understood and compared.

2.3.4 LEAF [24] is an open source ML benchmark suite for federated learning[24]. LEAF imitates a federated learning scenario, where models are trained on data produced by Edge devices in a decentralized fashion [20]. LEAF's data sets satisfy three conditions to represent realistic federated data sets: All data is keyed with an ID linking it to the device and user that produced it, generated in a large network of thousands to millions of devices and the distribution of data points is skewed across all devices in the network [24]. LEAF includes six data sets in total, five real world, and one synthetic data set. The real world data sets offer different sizes and number of devices. The workload associated to these data sets are image classification, sentiment analysis, classification, next-word, and next-character prediction[25]. LEAF also includes instructions for creating synthetic data sets, which let users specify the number of devices and is designed to create highly variable multi centered data sets [24]. Reference implementations using the federated learning algorithms SGD, FedAvg and Mocha are provided.

The metrics proposed in LEAF include measurements of the performance at the 10th, 50th and 90th percentile, an evaluation of the accuracy distribution across devices and data on the performance divided by the hierarchies in the data. This means that, for example for image classification on a set of pictures of handwritten characters, the accuracy across devices for each letter is measured. To not only capture a realistic picture of the performance distribution but to also measure the resources used on each device the number of FLOPS and bytes uploaded and downloaded on each device are included in the metrics as well.

2.3.5 CleanML [46] is a joint ML and data cleansing benchmark with an open source implementation[26]. It explores the influence of a set of error types and corresponding data cleaning methods on the quality of different ML models. CleanML provides 13 real world data sets with a combination of inconsistencies, duplicates, missing values and outliers, some of them with mislabeled data injected synthetically. For each error type one or more detection and repair methods are provided. It also specifies seven classification algorithms to train the models with: logistic regression, KNN, decision tree, random forest, Adaboost, XGBoost and naive Bayes.

The workloads are based on training on cleaned or dirty data and testing on cleaned data as well as testing a model trained on cleaned data with either cleaned or dirty test data. The effects of data cleaning on the quality of the model is evaluated for each combination of data set, data cleaning methods and model as well as for the best model and for the combination of the best cleaning method with the best model. A comparison of the accuracy or F1 score

[24] `https://github.com/TalwalkarLab/leaf`
[25] `https://github.com/TalwalkarLab/leaf`
[26] `https://github.com/chu-data-lab/CleanML`

of the different models determines if the model quality was affected positively, negatively or insignificantly by the data cleaning, this result is the main metric of CleanML.

3 Methodology

In this section we analyze the benchmarks presented in Section 2 under the light of four *dimensions*: *purpose*, *analytics pipeline stage*, *metrics*, and *convergence*, which compose a benchmarking *feature space*. Furthermore, we propose an *integrated data analytics* (IDA) architecture to illustrate the reach of current benchmarks and identify research opportunities.

3.1 Benchmarking Dimensions

Purpose relates to the aspects of the benchmark that will be used to stress the SUT. It is usually associated to data sets, application domain, kernels, or workloads. It is common that benchmarks have *multiple* purposes in order to stress a wider spectrum of characteristics of the SUT. Examples of purposes are a BD workload that performs external sorting on a large file [40], an HPC kernel that calculates matrix transposition [47], and a ML image classification application [49].

Stages refer to the different *stages* of the SUT that are stressed by the benchmark. As BD, HPC, and ML systems are often represented as *pipelines* composed by multiple stages, e.g., data processing, computation, and training, the pipeline stage dimension specifies which of those stages are stressed. For example, BigBench [36] is an *end-to-end* benchmark, stressing all levels (data collection, analysis, and storage) of the SUT in the data processing stage, while DeepBench [53] focuses only the training and inference stages of a ML pipeline.

Metrics are used to measure the performance of a SUT for given purposes and stages. For example, the FFT kernel of HPCC [47] is measured in PFLOP/s and stresses the computation stage, while BigBench's [36] MapReduce workloads are measured in seconds and stress the data processing stage.

Convergence defines the intersection between two or more of the benchmarking domains studied in this work, i.e., BD, HPC, and ML. That intersection may happen in the purpose or pipeline stage dimensions, or both, and reflect a trend observed on current BD, HPC, and ML systems. HPL-AI [5], for example, is at the convergence of HPC and ML since it has kernels based on both higher and lower precision floating pointing operations that replicate modern HPC and ML systems, respectively.

3.2 Integrated Data Analytics Pipelines

As discussed in Section 2, several benchmarks exist to assess BD, HPC, ML systems, and individual components. Figure 1 shows an abstraction of the ecosystem for an integrated data analytics (IDA) pipeline. This abstraction includes components from BD, HPC, ML systems. The stages of the IDA pipeline: *computation*, *data processing*, and *training*, correspond to the HPC, BD, and ML domains, respectively. We will use this abstraction in later section of this work to explore which of the existing benchmarks can cover multiple components of the three systems, and consequently, be used as a benchmark for such an IDA ecosystem.

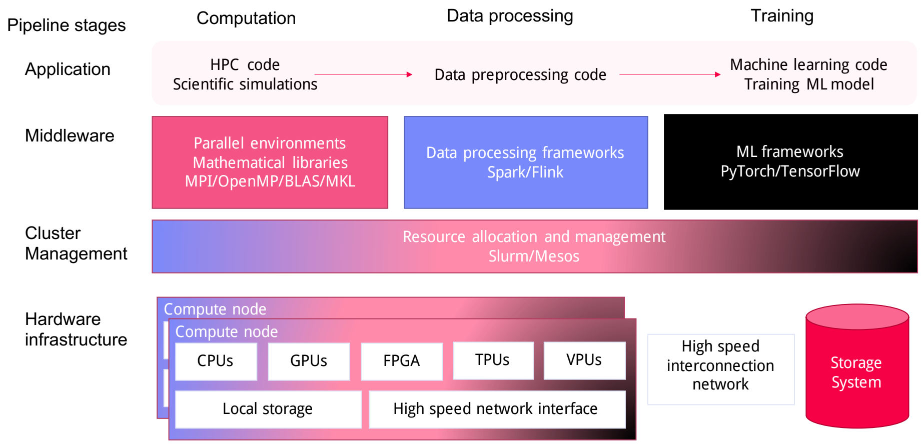

Fig. 1. Ecosystem for an integrated data analytics pipeline

3.3 Analysis of Big Data Benchmarks

BD systems commonly have three distinct steps: data collection, data analysis, and data storage. The *data analysis step* concentrates most efforts of current BD benchmarks, since that step is the focus of state-of-the-art BD systems, such as, stream processing, graph processing or MapReduce-based systems. Among the benchmark systems evaluated in Section 2.1, HiBench, BigBench, BigDataBench, and Graphalytics propose data analysis benchmarking.

The *data collection step* is explored by BigBench and Graphalytics, and, the *data storage step*, by BigBench and YCSB. Note that, while work like Graphalytics takes into consideration two of the three steps, end-to-end BD benchmarks, i.e., benchmark systems that consider all three data processing steps, such as BigBench, are still seldom in the literature.

In terms of *purpose*, the evaluated BD benchmark systems are all multi-purpose and very diverse, making them complementary. For the *metrics*, resource management and execution time are the most used, in particular for data analysis stages. For the data collection and storage levels, throughput and latency are also used.

Finally, the evaluated benchmarks describe some degree of convergence reflecting a trend observed in current production systems. Both BigBench and BigDataBench have ML purposes and benchmarking tasks based on performing statistical computations that simulate ML model training. The metrics related to those purposes, however, do not reflect ML performance.

Under the light of the Integrated Data Analytics pipeline (c.f., Section 3.2), the targets of the evaluated benchmark systems lie in the data processing pipeline stage and are part of Application and Middleware layers. In terms of converged benchmark systems, BigBench and BigDataBench, cover only the Application layer.

3.4 Analysis of High Performance Computing Benchmarks

HPC benchmarks usually focus on one or multiple pipeline stages which can involve parallel computation and data processing (including data communication and/or data storage). The parallel computation stage for HPC benchmarks follows the patterns given in the Berkeley Dwarfs [1]. Based on their computation pattern and workload, the benchmarks can involve one or more purposes such as targeting CPU, memory, I/O, network, etc.

From their purpose, we can also include the measurement metrics that each benchmark produces. Thus, we distinguish the benchmarks into the following categories: (i) *compute-centric benchmarks* measure floating point rate performance, testing compilers and runtime libraries, and testing code scalability and correctness, e.g., SPEC CPU 2017, HPCG/HPL, and NPB EP, (ii) *data-centric benchmarks* measure data movement, e.g., NPB ADC and the CORAL-2 Big Data benchmark which measures the data workload per sec, (iii) *network-centric benchmarks* measure communication performance between processing units, e.g., NPB FT and SPEC MPI, (iv) *memory-centric benchmarks* measure memory bandwidth, e.g., Coral-2 Memory Benchmark, HPCC STREAM, and UEABS NEMO, (v) *I/O-centric benchmarks*, e.g., CORAL-2 I/O Suite and NPB BTIO, and (vi) *a mix of the above categories* with overlapping problem interest, e.g. SPEC MPI.

We also observed certain research efforts towards benchmark convergence. HPL-AI highlights the emerging convergence of HPC and AI workloads. Unlike HPC benchmarks that mostly rely on 64-bit floating point operations accuracy, HPL-AI uses lower floating-point precision formats such as 16-bit which fit ML needs and offer better processing performance. Nevertheless, the metrics measured by these applications do not relate to ML accuracy but rather to measuring the execution time.

Another example is MLPerf's HPC benchmark[27], which is a step towards an end-to-end benchmark as the measured time does not include the training time only. Nevertheless, it captures the time of data staging from parallel file systems into accelerated and/or on-node storage systems.

[27] https://mlcommons.org/en/training-hpc-07/

3.5 Analysis of Machine Learning Benchmarks

ML systems are often composed of three main steps: data preparation, training, and inference. The purposes and metrics of those steps are usually disjoint, with data preparation involving purposes related to data cleaning and metrics such as execution time; training involves model training and time-to-accuracy related metrics; and inference involves model inference and time consumption metrics. This behavior is also observed in the evaluated benchmarking systems.

As the training step is usually the bottleneck of ML systems, we observe that all of the highlighted benchmark systems in Section 2.3 with exception of CleanML focus on the training step. They have multiple purposes spanning from matrix multiplication to sentiment analysis, but they all share time-to-accuracy and execution time metrics. DeepBench, in particular, also measures CPU consumption in teraflops and, together with MLPerf, has purposes associated to the inference stage. CleanML is the only of the evaluated benchmark systems that focuses on the preparation stage. It has purposes related to data cleaning, such as, duplicate detection, mislabeled data detection, and missing value correction. Its performance metric is the difference between the accuracy of a model training with data cleaned by the system under test and with cleaned data.

When analyzing the evaluated ML benchmarking systems in regards to the Integrated Data Analytics pipeline (c.f., Section 3.2), with the exception of DeepBench and MLPerf, the benchmarking systems mostly cover mostly the training part of Application and Middleware levels. DeepBench and MLPerf have purposes related to GPU training, hence, they also target the hardware infrastructure. CleanML also covers the data processing part of Integrated Data Analytics pipeline in spite of not considering the *volume* aspect of BD systems. Therefore, CleanML is the only of the evaluated benchmark systems to have convergence aspects.

4 Related Work

Many research efforts conducted extensive surveys to get deeper insights into the existing BD, HPC, and ML benchmarks. For instance, an early research effort introduced the Berkeley dwarfs [13], representing common computational patterns in HPC applications. These patterns span from linear algebra to graphical models, and finite state machine, and are important for understanding the breadth of the applications that a given benchmark can cover.

Another research effort introduced the BD ogres [30,31], which defines certain views/dimensions that capture common characteristics of BD applications. The views are not as simple as the computational patterns of the Berkeley dwarfs, with one view having many facets and covering several application characteristics.

The literature also comprises several surveys on benchmarks that focus on isolated stage levels (c.f., Section 3.1) of BD, HPC, and ML systems [67,28,21,16]. However, to the best of our knowledge, no other surveys focus on benchmarking of hybrid systems or systems covering two or more stages of the integrated data analysis pipeline (c.f., Section 3.2).

5 Conclusion

In this work, we discussed the state-of-the-art of BD, HPC, and ML benchmarks. We summarized a representative selection of some of the classic and most used benchmarks in the state of the art and classified them under the light of a feature space composed of purpose, stage, metric, and convergence, as well as from the perspective of a proposed Integrated Data Analysis architecture.

Through this classification, we observed that the evaluated benchmarking systems cover a wide range of purposes and stages of BD, HPC, and ML systems. However, even if modern hybrid-systems become more common, benchmarking systems are still not fully capable of targeting those data analytics systems.

We believe that a unified environment for benchmarking the performance of hybrid data analytics systems is a promising direction to explore in the future. We observed such behavior in a few benchmark systems. Nevertheless, we consider that such unified benchmarking environment could go *more broadly cover the three domains and more deeply capture the ecosystem layers* of the integrated data analysis (IDA) architecture, and hence, support *end-to-end converged benchmarking systems.*

Based on these observations, we argue that it would be possible to analyze features that are often ignored, in particular, middleware and software connections between systems from different domains (e.g., connection between the results of an HPC simulation and an ML training system). That position does not imply that "classic" kernel benchmarks are not needed, but that converged and end-to-end benchmark systems are needed and can open new dimensions for analyzing the performance of hybrid systems.

6 Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957407 as DAPHNE. This work has also been supported through the German Research Foundation as FONDA.

References

1. Computer architecture is back - the berkeley view on the parallel computing landscape. https://web.stanford.edu/class/ee380/Abstracts/070131-BerkeleyView1.7.pdf, accessed: 2021-08-18
2. Coral procurement benchmarks. https://asc.llnl.gov/sites/asc/files/2020-06/CORALBenchmarksProcedure-v26.pdf, accessed: 2021-06-30
3. High performance conjugate gradient benchmark (hpcg). https://github.com/hpcg-benchmark/hpcg/, accessed: 2021-07-04

4. High performance conjugate gradient benchmark (hpcg). http://www.netlib.org/benchmark/hpl/
5. Hpcg benchmark. https://icl.bitbucket.io/hpl-ai/, accessed: 2021-07-06
6. Parallel graph analytix (pgx). https://www.oracle.com/middleware/technologies/parallel-graph-analytix.html, accessed: 2021-07-01
7. Spec accel: Read me first. https://www.spec.org/accel/docs/readme1st.html#Q13, accessed: 2021-06-29
8. Spec omp 2012. https://www.spec.org/omp2012/, accessed: 2021-07-07
9. Specmpi. https://www.spec.org/mpi2007/, accessed: 2021-07-07
10. Standard performance evaluation corporation, spec cpu 2017. https://www.spec.org/cpu2017/Docs/overview.html#suites, accessed: 2021-06-29
11. Unified european applications benchmark suite. https://repository.prace-ri.eu/git/UEABS/ueabs, accessed: 2021-06-29
12. Adolf, R., Rama, S., Reagen, B., Wei, G.Y., Brooks, D.: Fathom: Reference workloads for modern deep learning methods. In: 2016 IEEE International Symposium on Workload Characterization (IISWC). pp. 1–10. IEEE (2016)
13. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., et al.: A view of the parallel computing landscape. Communications of the ACM **52**(10), 56–67 (2009)
14. Bailey, D., Barszcz, E., J.T, B., D.S, B., R.L, C., D, D., R.A, F., Frederickson, P., T.A, L., Schreiber, R., Simon, H., Venkatakrishnan, V., K, W.: The nas parallel benchmarks. , Technical report, RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 03 1994 (1994)
15. Bailey, D., Harris, T., Saphir, W., Wijngaart, R.v.d., Woo, A., Yarrow, M.: The nas parallel benchmarks 2.0. , Technical report, RNR-95-020, NASA Ames Research Center, Moffett Field, CA, 03 1995 (1995)
16. Bajaber, F., Sakr, S., Batarfi, O., Altalhi, A., Barnawi, A.: Benchmarking big data systems: A survey. Computer Communications **149**, 241–251 (2020). https://doi.org/https://doi.org/10.1016/j.comcom.2019.10.002, https://www.sciencedirect.com/science/article/pii/S0140366419312344
17. Barata, M., Bernardino, J., Furtado, P.: Ycsb and tpc-h: Big data and decision support benchmarks. In: 2014 IEEE International Congress on Big Data. pp. 800–801. IEEE (2014)
18. Baru, C., Bhandarkar, M., Curino, C., Danisch, M., Frank, M., Gowda, B., Jacobsen, H.A., Jie, H., Kumar, D., Nambiar, R., Poess, M., Raab, F., Rabl, T., Ravi, N., Sachs, K., Sen, S., Yi, L., Youn, C.: Discussion of bigbench: A proposed industry standard performance benchmark for big data. In: Nambiar, R., Poess, M. (eds.) Performance Characterization and Benchmarking. Traditional to Big Data. pp. 44–63. Springer International Publishing, Cham (2015)
19. Bienia, C., Kumar, S., Singh, J.P., Li, K.: The parsec benchmark suite: Characterization and architectural implications. In: Proceedings of the 17th international conference on Parallel architectures and compilation techniques. pp. 72–81 (2008)
20. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H.B., et al.: Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046 (2019)
21. Bonifati, A., Fletcher, G., Hidders, J., Iosup, A.: A Survey of Benchmarks for Graph-Processing Systems, pp. 163–186. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96193-4$_6$, https://doi.org/10.1007/978-3-319-96193-4_6

22. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1), 107–117 (1998). https://doi.org/https://doi.org/10.1016/S0169-7552(98)00110-X, https://www.sciencedirect.com/science/article/pii/S016975529800110X, proceedings of the Seventh International World Wide Web Conference
23. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems **30**(1-7), 107–117 (1998)
24. Caldas, S., Duddu, S.M.K., Wu, P., Li, T., Konečnỳ, J., McMahan, H.B., Smith, V., Talwalkar, A.: Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097 (2018)
25. Capotă, M., Hegeman, T., Iosup, A., Prat-Pérez, A., Erling, O., Boncz, P.: Graphalytics: A big data benchmark for graph-processing platforms. In: Proceedings of the GRADES'15, pp. 1–6 (2015)
26. Cheng, P., Lu, Y., Du, Y., Chen, Z.: Experiences of converging big data analytics frameworks with high performance computing systems. In: Asian Conference on Supercomputing Frontiers. pp. 90–106. Springer, Cham (2018)
27. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with ycsb. In: Proceedings of the 1st ACM symposium on Cloud computing. pp. 143–154 (2010)
28. Czarnul, P., Proficz, J., Krzywaniak, A., Weglarz, J.: Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. Sci. Program. **2019** (Jan 2019). https://doi.org/10.1155/2019/8348791, https://doi.org/10.1155/2019/8348791
29. Dongarra, J., Luszczek, P., Heroux, M.: Hpcg technical specification. Sandia National Laboratories, Sandia Report SAND2013-8752 (2013)
30. Fox, G.C., Jha, S., Qiu, J., Ekanazake, S., Luckow, A.: Towards a comprehensive set of big data benchmarks. Big Data and High Performance Computing **26**, 47 (2015)
31. Fox, G.C., Jha, S., Qiu, J., Luckow, A.: Ogres: a systematic approach to big data benchmarks. Big Data and Extreme-scale Computing (BDEC) pp. 29–30 (2015)
32. Frumkin, M.A., Shabanov, L.: Arithmetic data cube as a data intensive benchmark. , Technical report, NAS-03-005, NASA Ames Research Center, Moffett Field, CA, 03 2003 (2003)
33. Fuller, A., Fan, Z., Day, C., Barlow, C.: Digital twin: Enabling technologies, challenges and open research. IEEE Access **8** (2020)
34. Gao, W., Zhan, J., Wang, L., Luo, C., Zheng, D., Wen, X., Ren, R., Zheng, C., He, X., Ye, H., et al.: Bigdatabench: A scalable and unified big data and ai benchmark suite. arXiv preprint arXiv:1802.08254 (2018)
35. Gao, W., Zhu, Y., Jia, Z., Luo, C., Wang, L., Li, Z., Zhan, J., Qi, Y., He, Y., Gong, S., et al.: Bigdatabench: a big data benchmark suite from web search engines. arXiv preprint arXiv:1307.0320 (2013)
36. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., Jacobsen, H.A.: Bigbench: Towards an industry standard benchmark for big data analytics. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. p. 1197–1208. SIGMOD '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2463676.2463712
37. Guo, Y., Varbanescu, A.L., Iosup, A., Martella, C., Willke, T.L.: Benchmarking graph-processing platforms: A vision. In: Proceedings of the 5th ACM/SPEC international conference on Performance engineering. pp. 289–292 (2014)

38. Han, R., Zhan, S., Shao, C., Wang, J., John, L.K., Xu, J., Lu, G., Wang, L.: Bigdatabench-mt: A benchmark tool for generating realistic mixed data center workloads. In: BPOE. pp. 10–21. Springer (2015)
39. Huang, S., Huang, J., Dai, J., Xie, T., Huang, B.: The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In: 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010). pp. 41–51. IEEE (2010)
40. Huang, S., Huang, J., Liu, Y., Yi, L., Dai, J.: Hibench: A representative and comprehensive hadoop benchmark suite. In: Proc. ICDE Workshops. pp. 41–51 (2010)
41. Intel: Hibench (2021), https://github.com/Intel-bigdata/HiBench
42. Iosup, A., Hegeman, T., Ngai, W.L., Heldens, S., Prat-Pérez, A., Manhardto, T., Chafio, H., Capotă, M., Sundaram, N., Anderson, M., et al.: Ldbc graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. Proceedings of the VLDB Endowment **9**(13), 1317–1328 (2016)
43. Jack Dongarra, P.L.: HPC Challenge: Design, History, and Implementation Highlights, chap. 2. Chapman and Hall/CRC (2013)
44. Jack Dongarra, Mike Heroux, Piotr Luszczek: Bof hpcg benchmark update and a look at the hpl-ai benchmark
45. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. Journal of parallel and distributed computing **74**(7), 2561–2573 (2014)
46. Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., Zhang, C.: Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis]. arXiv preprint arXiv:1904.09483 p. 75 (2019)
47. Luszczek, P., Dongarra, J., Koester, D., Rabenseifner, R., Lucas, B., Kepner, J., McCalpin, J., Bailey, D., Takahashi, D.: Introduction to the hpc challenge benchmark suite (12 2004)
48. M. Dixit, K.: Overview of the spec benchmark. In: Gray, J. (ed.) The Benchmark Handbook, 1993, chap. 10, pp. 266–290. Morgan Kaufmann Publishers Inc. (1993)
49. Mattson, P., Cheng, C., Coleman, C., Diamos, G., Micikevicius, P., Patterson, D., Tang, H., Wei, G.Y., Bailis, P., Bittorf, V., et al.: Mlperf training benchmark. arXiv preprint arXiv:1910.01500 (2019)
50. Mattson, P., Reddi, V.J., Cheng, C., Coleman, C., Diamos, G., Kanter, D., Micikevicius, P., Patterson, D., Schmuelling, G., Tang, H., et al.: Mlperf: An industry standard benchmark suite for machine learning performance. IEEE Micro **40**(2), 8–16 (2020)
51. Ming, Z., Luo, C., Gao, W., Han, R., Yang, Q., Wang, L., Zhan, J.: Bdgs: A scalable big data generator suite in big data benchmarking. In: Advancing big data benchmarks, pp. 138–154. Springer (2013)
52. Müller, M., Whitney, B., Henschel, R., Kumaran, K.: SPEC Benchmarks, pp. 1886–1893. Springer US, Boston, MA (2011)
53. Narang, S.: Deepbench. https://svail.github.io/DeepBench/, accessed: 2021-07-03
54. Narang, S., Diamos, G.: An update to deepbench with a focus on deep learning inference. https://svail.github.io/DeepBench-update/, accessed: 2021-07-03
55. Ngai, W.L., Hegeman, T., Heldens, S., Iosup, A.: Granula: Toward fine-grained performance analysis of large-scale graph processing platforms. In: Proceedings of the Fifth International Workshop on Graph Data-management Experiences & Systems. pp. 1–6 (2017)
56. Poess, M., Nambiar, R.O., Walrath, D.: Why you should run tpc-ds: A workload analysis. In: Proceedings of the 33rd International Conference on Very Large Data Bases. p. 1138–1149. VLDB '07, VLDB Endowment (2007)

57. Rabl, T., Frank, M., Sergieh, H.M., Kosch, H.: A data generator for cloud-scale benchmarking. In: Nambiar, R., Poess, M. (eds.) Performance Evaluation, Measurement and Characterization of Complex Systems. pp. 41–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
58. Radulović, M., Asifuzzaman, K., Carpenter, P., Radojkovic, P., Ayguadé, E.: HPC Benchmarking: Scaling Right and Looking Beyond the Average: 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27 - 31, 2018, Proceedings, pp. 135–146 (01 2018). https://doi.org/10.1007/978-3-319-96983-1_10
59. Reed, D.A., Dongarra, J.: Exascale computing and big data. Communications of the ACM **58**(7), 56–68 (2015)
60. von Rueden, L., Mayer, S., Sifa, R., Bauckhage, C., Garcke, J.: Combining machine learning and simulation to a hybrid modelling approach: Current and future directions. In: Berthold, M.R., Feelders, A., Krempl, G. (eds.) Advances in Intelligent Data Analysis XVIII. pp. 548–560. Springer International Publishing, Cham (2020)
61. Tian, X., Dai, S., Du, Z., Gao, W., Ren, R., Cheng, Y., Zhang, Z., Jia, Z., Wang, P., Zhan, J.: Bigdatabench-s: An open-source scientific big data benchmark suite. In: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 1068–1077. IEEE (2017)
62. Vazhkudai, S.S., de Supinski, B.R., Bland, A.S., Geist, A., Sexton, J., Kahle, J., Zimmer, C.J., Atchley, S., Oral, S., Maxwell, D.E., Larrea, V.G.V., Bertsch, A., Goldstone, R., Joubert, W., Chambreau, C., Appelhans, D., Blackmore, R., Casses, B., Chochia, G., Davison, G., Ezell, M.A., Gooding, T., Gonsiorowski, E., Grinberg, L., Hanson, B., Hartner, B., Karlin, I., Leininger, M.L., Leverman, D., Marroquin, C., Moody, A., Ohmacht, M., Pankajakshan, R., Pizzano, F., Rogers, J.H., Rosenburg, B., Schmidt, D., Shankar, M., Wang, F., Watson, P., Walkup, B., Weems, L.D., Yin, J.: The design, deployment, and evaluation of the coral pre-exascale systems. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 661–672 (2018)
63. Walter Lioen, Miguel Avillez, V.C.D.D.S.D.A.E.J.F.C.J.M.L.C.M.C.M.A.P.A.S.: Evaluation of accelerated and non-accelerated benchmarks. (2019)
64. Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., et al.: Bigdatabench: A big data benchmark suite from internet services. In: 2014 IEEE 20th international symposium on high performance computer architecture (HPCA). pp. 488–499. IEEE (2014)
65. Wijngaart, R.v.d., Jin, H.: Nas parallel benchmarks, multi-zone versions. , Technical report, NAS-03-010, NASA Ames Research Center, Moffett Field, CA, 03 2003 (2003)
66. Wong, P., Wijngaart, R.v.d.: Nas parallel benchmarks i/o version 2.4. , Technical report, NAS-03-020, NASA Ames Research Center, Moffett Field, CA, 03 2003 (2003)
67. Zhang, Q., Zha, L., Lin, J., Tu, D., Li, M., Liang, F., Wu, R., Lu, X.: A survey on deep learning benchmarks: Do we still need new ones? In: Zheng, C., Zhan, J. (eds.) Benchmarking, Measuring, and Optimizing. pp. 36–49. Springer International Publishing, Cham (2019)