ABench: Big Data Architecture Stack Benchmark

[Vision Paper]

Todor Ivanov
Frankfurt Big Data Lab, Goethe University
Frankfurt am Main, Germany
todor@dbis.cs.uni-frankfurt.de

Rekha Singhal
TCS Research
Mumbai, India
rekha.singhal@tcs.com

ABSTRACT

Distributed big data processing and analytics applications demand a comprehensive end-to-end architecture stack consisting of big data technologies. However, there are many possible architecture patterns (e.g. Lambda, Kappa or Pipeline architectures) to choose from when implementing the application requirements. A big data technology in isolation may be best performing for a particular application, but its performance in connection with other technologies depends on the connectors and the environment. Similarly, existing big data benchmarks evaluate the performance of different technologies in isolation, but no work has been done on benchmarking big data architecture stacks as a whole. For example, BigBench (TPCx-BB) may be used to evaluate the performance of Spark, but is it applicable to PySpark or to Spark with Kafka stack as well? What is the impact of having different programming environments and/or any other technology like Spark? This vision paper proposes a new category of benchmark, called ABench, to fill this gap and discusses key aspects necessary for the performance evaluation of different big data architecture stacks.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **Information systems** → *Data management systems*; • **Computing methodologies** → *Symbolic and algebraic manipulation*;

KEYWORDS

ABench, BigBench, Big Data Benchmarking, Big Data

ACM Reference Format:
Todor Ivanov and Rekha Singhal. 2018. ABench: Big Data Architecture Stack Benchmark: [Vision Paper]. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3185768.3186300

1 MOTIVATION

There is a growing number of new applications and use cases that challenge the capabilities of the existing systems due to availability of large size and high speed data. Big data analytics is one of the applications which is going viral both inside and outside an enterprise. The 3Vs characteristics of Big Data are still changing their dimensions helped by new Vs, defined by the growing need of the industry. On the other hand, the speed with which new technologies and features are emerging makes it very hard for both developers and users to keep track of the best technology on the market. Nowadays, many of these software tools are open source and created by communities, which do not have sufficient time and resources to keep the documentation up-to-date and present the tool in a way comparable to the enterprise products. Moreover, full deployment of an application may require multiple technologies connected to each other such as streaming applications which may be deployed on Kafka and Storm platforms. This architecture should be modular by design, to accommodate mix-and-match configuration options as they arise. Lambda and Kappa architectures can differ in implementation, depending on technology deployed at each of the layers in the architecture stack. Exactly these variations in the architectural components lead to the relevance of the heterogeneity in big data architectures. It is a concept outlined in multiple studies [7, 8, 13] as an emerging trend, which will open many new challenges. Moreover, size of application workload and data size may play a critical role in choosing an architecture. Performance prediction models such as [10, 11] may be extended for architecture benchmarks to predict the performance for different workload and data size avoiding multiple executions of the benchmark.

There is a need to benchmark the performance of a given solution architecture which may get deployed using different set of technologies and hardware. We propose a new type of benchmark to evaluate the performance of big data stacks for different deployment architectures. This benchmark can act as a tool to evaluate performance of a particular big data technology/architecture stack for desired hardware. The benchmark can be used by solution architects to compare features and performance of an architecture instances with different technology specified at each layer. The available benchmarks [12] (TPC-DS, TPCx-BB, TPCx-HS, etc.) measure performance of a technology but do not address the performance of connectors connecting two technologies for creating an architecture.

The reminder of the paper is organized as follows: Section 2 gives an overview of the proposed benchmark and presents the main benchmark concepts. Section 3 presents a general benchmark framework for big data architecture benchmarks with challenges. Section 4 presents two use cases with BigBench and the final Section 5 concludes the paper.

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/10.1145/3185768.3186300

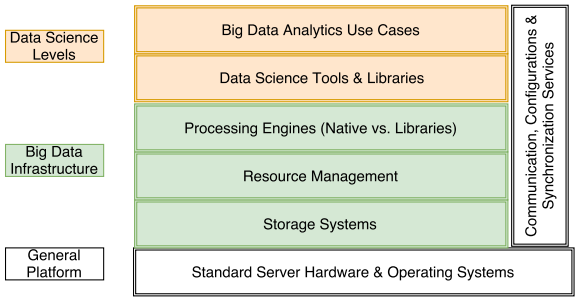

Figure 1: Abstract Big Data Stack

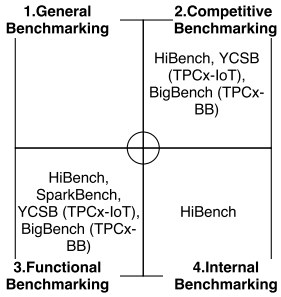

Figure 2: Types of Benchmarks

2 BENCHMARK OVERVIEW

Historically, the TPC benchmarks [12] have been used as an industry standard for performance comparisons of hardware and software systems. The TPC benchmark specifications are implemented by the vendors and then audited for transparency and fairness purposes. Today, several new technologies are open source and developed by communities such as the Apache Foundation and not by a single vendor company. Similarly, the primary users of the emerging benchmarks are the data engineers, software developers and architects participating in this open source communities, in contrast to the declining number of enterprises ready to create their own implementation of a TPC benchmark, execute it and finally go through the process of result auditing. For example, the TPC-DS benchmark is often used to stress test the SQL-on-Hadoop engines such as Hive, Impala and SparkSQL, but no officially audited results are published.

At the same time, the variety of new emerging big data technologies (including Data Science, Machine Learning and Deep Learning tools) opens the space and need for new standardized benchmarks that target exactly these new tools and analytics techniques. The challenge is to find new methodologies and techniques that will address this problem and provide us with a practical approach that solves the benchmarking gap. Our solution to the benchmarking gap is to develop a new type of Big Data Architecture Stack benchmark (ABench) that will first incorporate the best practices of the existing benchmarks and second will try to use innovative approaches to solve the challenges posed by the new big data technologies. In contrast to the typical TPC benchmarks, which provide strict specification, ABench will be more similar to the Java Client/Server benchmark defined by SPEC. The benchmark framework will cover a broad spectrum of realistic use cases and the common best practice architectures for implementing them.

As mentioned, one of the biggest developers' challenge is to deal with the complexity and variety of big data technologies in all layers of the data platform stacks. Figure 1 depicts the functional layers in a typical big data platform in an abstract way. Implementing a typical big data use case involves the use of tools from most of the depicted layers in the stack. This requires that the application developer has a good knowledge first of the application requirements and second of the available big data technologies at each layer. After identifying the necessary tools, he/she should be able to configure them properly and make sure they can exchange data in an effective way. The platform administration and the optimal configuration are other important points that are key for the application performance. What will be helpful in this situation? As a benchmark framework, ABench will provide a common big data application implementation that can be used for platform testing and starting point in the development process. For example, for a streaming application using a pre-configured Lambda or Kappa architecture with the implemented tools on the different stack layers will immensely reduce the starting overhead for the developer. Meanwhile, setting up the benchmark on his infrastructure will enable him to directly measure the performance of both the single tool and the entire platform stack including the overhead for data exchanges between the components. Furthermore, making our benchmark framework open source and accessible to everyone will give the opportunity for the users to contribute their code and improve the performance of the implemented best practices. Some of the basic principle that will be followed are:

- Open source implementation and extendable design
- Easy to setup and extend
- Include data generator or public data sets to simulate workload that stresses the architecture
- Reuse of existing benchmarks

Another important aspect in our vision is to define the benchmarking perspectives of ABench. Inspired by Andersen and Pettersen [1], who defined four benchmark types (generic, competitive, functional and internal benchmarking) in the context of a company comparison, we adapt these four benchmark types to the context of complex big data architecture stacks as follows:

- **Generic Benchmarking** checks if the general business requirements and specifications according to which the implementation is done are fulfilled.
- **Competitive Benchmarking** is a performance comparison between the best tools on the platform layer that offer similar functionality.
- **Functional Benchmarking** is a functional comparison of the features of the tool against technologies from the same area.
- **Internal Benchmarking** is on the lowest level, comparing variations of the implementation code done using a particular tool.

Figure 2 briefly depicts the four types of benchmarks in quadrants together with examples of popular big data benchmarks. The generic benchmarking is done manually by people and as such it is hard

to automate by software. However, it is possible to classify the big data applications in categories according to industry sector and domain, so that each category is represented by a benchmark that covers all main business requirements and specifications. Currently, many popular big data benchmarks such as HiBench [6], YSCB (TPCx-IoT) [2] and BigBench (TPCx-BB) [4, 5] are typically used for competitive and functional benchmarking to compare common features of different big data technologies. They focus on testing the technical aspects of big data technologies and are built with different goals in mind. For example, BigBench covers competitive (Hive, SparkSQL, etc.) and functional (the same HiveQL code on Hive and SparkSQL) benchmarking, SparkBench targets only the Spark engine and as such covers only the functional benchmarking, whereas HiBench covers competitive, functional (Spark, Flink, etc.) and internal benchmarking (Scala, Java, Python).

In short, our goal is to cover all four benchmark types and develop ABench as a multi-purpose benchmark framework that can be used in many big data scenarios. For example, one approach is to extend BigBench to cover more general application scenarios (e.g. streaming, machine learning, etc. for generic benchmarking requirements), to provide implementations in multiple big data technologies (e.g. Flink, Kafka, Impala etc. for competitive and functional benchmarking), and to offer implementations in different APIs (e.g. Spark, PySpark, R, etc. for internal benchmarking).

3 ARCHITECTURE BENCHMARK FRAMEWORK

Our proposal for the ABench benchmark framework shall stress test the common application business requirements (e.g. retail analytics, retail operational, etc.), big data technologies functionalities and best practice implementation architectures. The benchmark framework need to define following components:

3.1 Data Model

The types of data models and relationship across them. For example, a retail application may have structured, unstructured, semi-structures and stream data types and a social networking application may have graph data types. The relationship across different types of data need to be specified on how the one is derived from another. For relational and key-value store, data access mechanism need to be defined - sequential or random scan through index.

3.2 Data Store

For each type of data model, possible actions on data model with their required performance need to be specified. This may be used to decide the storage type in terms of persistence, partitioning, in-memory, duplicates or combination of these.

3.3 Data Generation

The framework need to have different types of data generators for stream messages, structured, graph, unstructured, documents etc. For example, stream data generator may also need to specify number of streams and velocity of generation whereas graph generator may specify number of nodes and edges between them. Depending on the data models in the architecture, the generators shall capture their interdependency.

3.4 Workload

The workloads could be business problem dependent if only specifications are given or could be in form of SQL queries which may access data across different data models. The workloads could include graph queries, operational system workload, machine learning analytics, continuous queries and stream analytics. Each use case will focus on different functionality and system architecture, which also means that it will be implemented in different technologies.

3.5 Data Consistency and Security

Application level data consistency requirements such as ACID, CAP, etc. impact the performance of a technology. Moreover, different security constraints with different technology may result in different performance. For example mature technologies may ensure strict security versus recent ones. The workloads shall be able to capture these aspects in the benchmark.

3.6 Benchmark Control Knobs

The benchmark shall be able to execute with varying number of concurrent users per second whereby each user may click with his/her own speed (including think time) which results in the total concurrent sessions supported in the system. For example, in a streaming case each web click will generate a message which may also be controlled since the performance of messages and connectors depend on the message size moving across different layers of the architecture stack. The data sizes shall be varying for all data types - should we have control on feeding skewness in the data sets especially in web clicks, where a user may click only one type of products repeatedly.

3.7 Performance Data Collection

The benchmark shall have mechanism to collect technology specific, component specific and architecture specific performance counters.

3.8 Benchmark Metric

Defining a benchmark metric (e.g. TPS or BBQpm@SF) is always a challenging task and especially in complex environments. Our benchmark defines functionally independent components (stack layer implemented in particular technology), which logically will have different internal metrics and measures. However, for all components the execution time is one of the most important metrics that we adapt as a main metric in the benchmark. The total (End-to-end) execution time of an application scenario is the sum of the execution times of each benchmark component including the technology and its connector. The other most important performance metric especially for big data distributed system are scalability and reliability. One may need to define and calculate these in the context of an architecture either at each component level or the whole stack. The benchmark could have other metrics as throughput, energy efficiency and cost of the solution as well.

3.9 Challenges

Below are the key challenges which one need to address while designing and/or creating architecture benchmarks.

- An architecture benchmark may be defined as hybrid of available benchmarks. The challenge may be to make those benchmarks work together on the same platform and build connectors across the chosen benchmarks
- Benchmark specifications may be provided to capture all functional and non-functional details (e.g. people, time, etc.). The challenge may be to give sufficient details for vendors to implement on any given set of technologies. Also, the validation of their implementations and result will be difficult.
- Express benchmarks (as defined by TPC) for a few architecture patterns with multiple technologies. However, it requires a lot of implementations and standardization of these benchmarks across different deployment systems.
- Where data generators should be residing especially for the dynamic data generator (such as stream), which may interfere with benchmark performance?
- Can the benchmark materialize fewer data sets in between to improve performance?

4 BENCHMARK USE CASES

This section proposes two concrete use cases for architecture benchmarks (as part of ABench) by re-using BigBench [4, 5] as a baseline. The goal is to extend its retail business scenario to cover both stream processing workloads and advanced machine learning techniques.

4.1 Stream Processing

A retail business application deployment may need message platform (e.g. Kafka), streaming engine (e.g. Spark, Storm, Flink), in-memory store (e.g. MemSQL, MongoDB) and persistent data store (HDFS, HBase). A benchmark is needed to evaluate the performance of the whole architecture stack. The benchmark shall have control on ingestions of workloads in terms of the number of concurrent sessions and data size. Data streaming and processing is one type of workload which is still not addressed in BigBench and currently represents a huge interests in both research and industry. Moreover, continuous query workloads are not benchmarked with the current BigBench. Therefore, there are multiple system architectures like the Lambda and Kappa architectures that can be used as an implementation standards for this type of applications.

The stream dataset size is a function of the number, size and rate of messages per second. Semi-structured and structured datasets are function of the data size in the file or table. Some streaming queries in BigBench are:

- Find top 10 products (or categories) that are viewed by most of the (at least 50 customers viewed that) users in last 10 minutes from current date and time.
- Generate an offer if a user has done total purchase of more than USD 1000 together in his last 5 transactions.
- Find the top selling 10 products in last one hour.
- Show the number of unique visitors in last one hour.

4.2 Machine Learning

The traditional descriptive analytics and business intelligence (BI) have evolved and companies today rely on various machine learning (ML) techniques to get better and faster business insights. Gartner [3] has defined four types of advanced analytics that businesses adapt: **descriptive analytics**, **diagnostic analytics**, **predictive analytics** and **prescriptive analytics**.

In the current BigBench [4, 5], five (Q5, Q20, Q25, Q26 and Q28) out of the 30 queries are covering common ML algorithms like Clustering (K-Means) or Classification (Logistic Regression and Naive Bayes). A recent evaluation of the benchmark has proposed to extend the BigBench workload with Collaborative Filtering using Matrix Factorization implementation in Spark MLlib via the Alternating Least Squares (ALS) method [9]. The main objective is to extend the existing workloads to cover wider spectrum of the four advanced analytics types. At the same time there is a need of new type of ML metrics that will allow to compare the scalability and accuracy of different ML frameworks.

5 CONCLUSIONS

In this paper, we have proposed a new type of multi-purpose benchmark framework for big data architecture stacks, called ABench. It can be created reusing or extending existing big data benchmarks such as Hibench and BigBench. We have outlined a framework for these new benchmarks and proposed streaming and machine learning extensions based on BigBench.

ACKNOWLEDGMENTS

This research has been supported by the Research Group of the Standard Performance Evaluation Corporation (SPEC). Special thanks for the valuable feedback to Roberto V. Zicari (Frankfurt Big Data Lab).

REFERENCES

[1] Bjørn Andersen and P-G Pettersen. 1995. *Benchmarking handbook.* Champman & Hall.
[2] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010.* 143–154.
[3] Gartner. 2017. Planning Guide for Data and Analytics. www.gartner.com/doc/3471553/-planning-guide-data-analytics. (2017).
[4] Ahmad Ghazal, Todor Ivanov, Pekka Kostamaa, Alain Crolotte, Ryan Voong, Mohammed Al-Kateb, Waleed Ghazal, and Roberto V. Zicari. 2017. BigBench V2: The New and Improved BigBench. In *ICDE 2017, San Diego, CA, USA, April 19-22.*
[5] Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. 2013. BigBench: Towards An Industry Standard Benchmark for Big Data Analytics. In *SIGMOD 2013.* 1197–1208.
[6] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. 2010. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on.* IEEE, 41–51.
[7] Todor Ivanov, Sead Izberovic, and Nikolaos Korfiatis. 2016. The Heterogeneity Paradigm in Big Data Architectures. In *Managing and Processing Big Data in Cloud Computing.* IGI Global, 218–245.
[8] Sankaralingam Panneerselvam and Michael Swift. 2016. Rinnegan: Efficient Resource Use in Heterogeneous Architectures *(PACT 2016).* ACM, New York, USA, 373–386.
[9] Sweta Singh. 2016. Benchmarking Spark Machine Learning Using BigBench. In *8th TPC Technology Conference, TPCTC 2016, New Delhi, India, September 5-9, 2016.*
[10] Rekha Singhal and Praveen Singh. 2017. Performance Assurance Model for Applications on Spark Platform. In *9th TPC Technology Conference 2017.*
[11] Rekha Singhal and Abhishek Verma. 2016. Predicting Job Completion Time in Heterogeneous MapReduce Environments. In *IPDPS Work. 2016, Chicago, USA, May 23-27.*
[12] TPC. 2018. www.tpc.org/. (2018).
[13] Dongyao Wu, Liming Zhu, Xiwei Xu, Sherif Sakr, Daniel Sun, and Qinghua Lu. 2016. Building Pipelines for Heterogeneous Execution Environments for Big Data Processing. *IEEE Softw.* (2016), 8.