# BigDataBench-MT: A Benchmark Tool for Generating Realistic Mixed Data Center Workloads

[1]Rui Han, [3]Shulin Zhan, [4]Chenrong Shao, [5]Junwei Wang,
[6]Lizy K. John, [1]Jiangtao Xu, [1,2]Gang Lu, [1]Lei Wang

[1]Institute Of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, China
[3]iCarsclub, Beijing, China
[4]Xi'an Jiaotong University, Xi'an, China
[5]Kingsoft Cloud, Beijing, China
[6]Department of Electrical and Computer Engineering, the University of Texas at Austin, TX, USA
hanrui@ict.ac.cn, zhanshulin@ppzuche.com, scr1994@stu.xjtu.edu.cn, wangjunwei@kingsoft.com, ljohn@ece.utexas.edu, {xujiangtao, lugang, wl}@ict.ac.cn

**Abstract.** Long-running service workloads (e.g. web search engine) and short-term data analysis workloads (e.g. Hadoop MapReduce jobs) colocate in today's data centers. Developing realistic benchmarks to reflect such practical scenario of mixed workload is a key problem to produce trustworthy results when evaluating and comparing data center systems. This requires using actual workloads as well as guaranteeing their submissions to follow patterns hidden in real-world traces. However, existing benchmarks either generate actual workloads based on probability models, or replay real-world workload traces using basic I/O operations. To fill this gap, we propose a benchmark tool that is a first step towards generating a mix of actual service and data analysis workloads on the basis of real workload traces. Our tool includes a combiner that enables the replaying of actual workloads according to the workload traces, and a multi-tenant generator that flexibly scales the workloads up and down according to users' requirements. Based on this, our demo illustrates the workload customization and generation process using a visual interface. The proposed tool, called BigDataBench-MT, is a multi-tenant version of our comprehensive benchmark suite BigDataBench and it is publicly available from http://prof.ict.ac.cn/BigDataBench/multi-tenancyversion/.

**Keywords:** data center; benchmark; workload trace; mixed workloads

## 1 Introduction

In modern cloud data centers, a large number of tenants are consolidated to share a common computing infrastructure and execute a diverse mix of workloads.

Benchmarking and understanding these workloads is a key problem for system designers, programmers and researchers to optimize the performance and energy efficiency of data center systems and to promote the development of data center technology. This work focuses on two classes of popular data center workloads [46]:

- *Long-running services.* These workloads offer online services such as web search engines and e-commerce sites to end users and the services usually keep running for months and years. The *tenants* of such workloads are *service end users.*
- *Short-term data analysis jobs.* These workloads process input data of many scales using relatively short periods (e.g. in Google and Facebook data centers, a majority (over 90%) of analytic jobs complete within a few minutes [30,46]). The *tenants* of such workloads are *job submitters.*

As data analysis systems such as Hadoop and Spark mature, both types of workloads widely co-locate in today's data centers, hence the pressure to benchmark and understand these mixed workloads rises. Within this context, we believe that it will be of interest to the data management community and a large user base to generate realistic workloads such that trustworthy benchmarking reflecting the practical data center scenarios can be conducted. Considering the heterogeneity and dynamic nature of data center workloads and their aggregated resource demands and arrival patterns, this requires overcoming two major challenges.

**Benchmarking using actual workloads based on real-world workload traces**. Data analysis jobs usually have various computation semantics (i.e. implementation logics or source codes) and input data sizes (e.g. ranging from KB to PB), and their behaviors also heavily rely on the underlying software stacks (such as Hadoop or MPI). Hence it is difficult to emulate the behaviors of such highly diverse workloads just using synthetic workloads such as I/O operations. On the other hand, generating workloads whose arrival patterns follow real-world traces is an equally important aspect of realistic workloads. This is because these traces are the most realistic data sources including both explicit and implicit arrival patterns (e.g. sequences of time stamped requests or jobs).

**Benchmarking using scalable workloads with realistic mixes**. A good benchmark needs to flexibly adjust the scale of workloads to meet the requirements of different benchmarking scenarios. Based on our experience, we noticed that in many cases, obtaining real workload traces is difficult due to confidential issues. The limited trace data also restrict the scalability of benchmark. It is therefore challenging to produce workloads at different scales while still guaranteeing their realistic mix corresponding to real-world scenarios.

In this paper, we propose a benchmark tool that is a first step towards generating realistic mixed data center workloads. This tool, called BigDataBench-MT, is a multi-tenancy version of our open-source project BigDataBench, which is a comprehensive benchmark suite including 14 real-world data sets and 33 actual workloads covering five application domains [8,51]. The goal of BigDataBench-MT is not only supporting the generation of service and data analysis work-

loads based on real workload traces, but also providing a multi-tenant framework to enable the scaling up and down of such workloads with guarantee of their realistic mixes. Considering our community may feel interest in using these workloads to evaluate new system designs and implementations, our tool and the corresponding workload traces are publicly available from `http://prof.ict.ac.cn/BigDataBench/multi-tenancyversion/`.

## 2 Related Work

We now review existing data center benchmarks from three perspectives, as shown in Table 1.

**Evaluated platform**. First of all, we classify data center benchmarks according to their targeted systems. We consider three popular camps of systems in today's data centers: (1) *Hadoop-related systems*: the great prosperity of the Hadoop-centric systems in industry brings a wide diversity of systems (e.g. Spark [6], HBase[1], Hive [2] and Impala [3]) on top of Hadoop MapReduce and HDFS [11] as well as a wide range of benchmarks specifically designed for these systems. (2) *Data stores*: parallel DBMSs (e.g. MySQL [12] and Oracle [15]) and NoSQL data stores (e.g. Amazon Dynamo [33], Cassandra [41] and Linkedin Voldemort [50]) also widely exist in data centers. (3) *Web services*: long-running web services such as Nutch search engine [5] and multi-tier cloud applications [36] are another important type of data center applications. These services usually have stringent response time requirement [37] and their request processing is distributed into a large number of service components for parallel processing, thus the service latency is determined by the tail latency of these components [32,38].

**Workload implementation logic**. Consider the complexity and diversity of workload behaviors in current data center systems, the implementation logic of existing data center benchmarks can be classified into three categories. The first category of benchmarks implement their workloads with algorithms. For example, HiBench [39] include workloads implemented with machine learning algorithms in Mahout [4]. The second category of benchmarks implement workloads using database operations such as reading, loading, joining, grouping, unifying, ordering, aggregating and spliting data. The third category of benchmarks implement workloads as I/O operations. For example, NNBench [13] and TestDFSIO [23] emulate I/O operations on Hadoop HDFS; GridMix [10] provides two workloads: LoadJob that performs I/O operations and SleepJob that sleeps the jobs; and SWIM [30] provides four workloads that stimulate the operations of Hadoop jobs to read, write, shuffle and sort data. We view the first two categories of workloads as *actual workloads*, because these workloads have semantics and they consume resources of processors, memories, caches and I/O bandwidths in execution. By contrast, workloads belonging to third category only consume I/O resources.

**Workload mix**. Finally, we classify data center benchmarks into three categories from the perspective of workload mix. The first type of data center

**Table 1.** Overview of data center benchmarks

| **Workload mix** | **Workload implementation logic** | | |
|---|---|---|---|
| | **Algorithms** | **Database operations** | **I/O operations** |
| **No mix** | WordCount[1][26], Grep[1][9], Sort[1][19], Terasort[1][22], HiBench[1][39], TPCx-HS[1][43], Graphalytics[1][29], CloudSuite[4][34] | MRBench[1][40], CALDA[2][44], AMPLab benchmark[2][7], YCSB[2][31], BG benchmark[2][28], CloudSuite[4][34] | NNBench[1][13], TestDFSIO[1][23], HiBD[1][49] |
| **Synthetic mix** | HcBench[1][47], MRBS[1][48] | PigMix[1][17], HcBench[1][47], MRBS[1][48], BigBench[2][35], LinkBench[2][27], TPC benchmarks[2][24], TPC-W[3][25], BigDataBench[4][51] | HiBench[1][39], SPECWeb99[3][20] |
| **Realistic mix** | | | Gridmix[1][10], SWIM[1][21,30] |

[1] Hadoop-related systems
[2] Data stores
[3] Web services
[4] All three types of systems

benchmarks either generate single workloads (e.g. WordCount [26], Grep [9] and Sort [19]) or generate multiple workloads individually (e.g. CALDA [44], AMPLab benchmark [7] and CloudSuite [34]). That is, these benchmarks donot consider workload mix. The second category of benchmarks generate synthetic mixes of workloads. Many benchmarks (e.g. PigMix [17], HcBench [47] and BigBench [35]) generate mixes of workloads by manually determining their proportions. Similarly, TPC benchmarks [24] design a query set as a synthetic mix of queries with different proportions. YCSB [31] uses a package to include a set of related workloads. MRBS decides the frequencies of different workloads using probability distributions such as a random distribution. Finally, third category of benchmarks generate a realistic mix of synthetic workloads whose arrival patterns faithfully follow real-world traces. For example, GridMix [10] and SWIM [21,30] first build a job trace to describe the realistic job mix by mining production loads, and then run synthetic I/O operations according to the trace. However, how to generate actual workloads on the basis of real workload traces is still an open question.

## 3 System Overview

Figure 1 shows the framework of our benchmark tool. It consists of three main modules. In the *Benchmark User Portal*, users can first specify their benchmarking requirements, including the machine type and number to be tested, and the types of workload to use. A set of workload traces following these requirements are then selected. The next step of *Combiner of Workloads and Traces* is to

match the real workload and the selected workload traces, and outputs workload replaying scripts to guide the workload generation. Finally, the *Multi-tenant Workload Generator* extracts the tenant information from the scripts and constructs a multi-tenant framework to generate a mix of service and data analysis workloads.

In BigDataBench-MT, we employ the Sogou user query logs [18] as the basis to generate the service workload (i.e. the Nutch search engine [5]) and the Google cluster workload trace as the basis to generate data analysis workloads (i.e. Hadoop and Shark workloads). The Sogou trace records logs from 50 days and it includes over 9 million users and 43 million queries. The Google trace records logs from 29 days and 12,492 machines and it includes over 5K users, 40K workload types, 1000K jobs and 144 million tasks. As a preprocessing step, we converted both traces into Impala databases (full version) and MySQL database (24-hour version) to facilitate the customization of benchmarking scenarios. In the following subsections, we describe the last two modules of our tool.

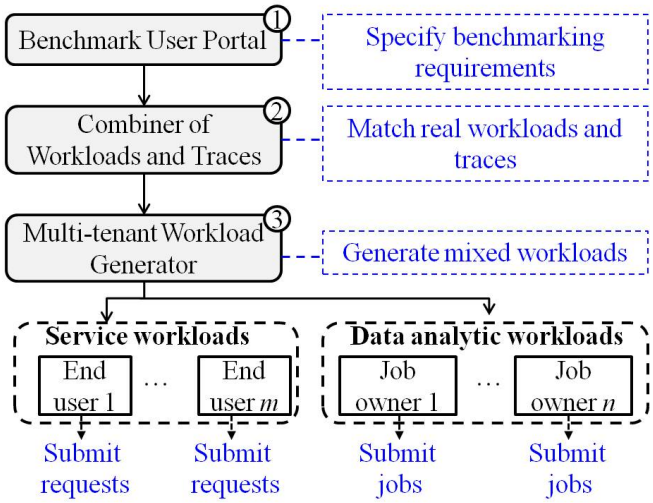


**Fig. 1.** The BigDataBench-MT framework

### 3.1 Combiner of Workload and Traces

The goal of the combiner is to extract the request/job arrival patterns from real-world traces and combine them with actual workloads. The combiner applies differentiated combination techniques to the service and data analysis workloads because their workload generations have different features.

**Service workloads**. The generation of a service workload is determined by three factors: the request submitting time, the sequence of requests and the

content of each request query. Take the web search engine for example, the combiner implements a request submitting service that automatically derives these factors from the Sogou trace and uses them to determine the request submission process.

**Data analysis workloads**. The generation of a data analysis workload is determined by four factors: the job submitting time, the workload type (i.e. the computation semantics and the software stack) and the input data (i.e. the data source and size). The current workload traces usually show the information of job submitting time but only provide *anonymous jobs* whose workload types and/or input data are unknown. Hence the basic idea of the combiner is to derive the workload characteristics of both actual and anonymous jobs and then match jobs whose workload characteristics are sufficiently similar. Table 2 lists the metrics used to represent workload characteristics, which reflect both jobs' performance (execution time and resource usage) and micro-architectural behaviors (CPI and MAI).

**Table 2.** Metrics to represent workload characteristics of data analysis jobs

| **Metric** | **Description** |
|---|---|
| Execution time | Measured in seconds |
| CPU usage | Total CPU time per second |
| Total memory size | Measured in GB |
| CPI | Cycles per instruction |
| MAI | The number of memory accesses per instruction |

Figure 2 shows the process of matching actual data analysis jobs and traces' anonymous jobs and it consists of two parallel sub-processes. First, the actual jobs with different input data sizes are tested and their metrics of workload characteristics are collected. In BigDataBench-MT, we provide auto-running scripts to collect performance metrics and hardware performance counters (Perf [16] and Oprofile [14] for Linux 2.6+ based systems) to obtain micro-architectural metrics. Using the testing results as samples, the combiner trains the multivariate regression model to describe the relationship between an actual job (including both its workload type and input size as the independent variables) and its workload characteristic metrics (one metric is a dependent variable). Second, the combiner views each anonymous job as an entity and the five workload characteristic metrics as its attributes, and employs the Bayesian Information Criterion (BIC)-based k-means clustering algorithm [45] to group anonymous jobs in the trace into different clusters.

Based on the constructed regression models and clusters, the combiner further matches each cluster to one actual job with a specific input data. In the matching, the coefficient of variation (CV) measure, defined as the ratio of the standard deviation $\sigma$ to the mean $\mu$, is used to describe the dispersion of jobs in the same cluster. The combiner iteratively tests actual jobs of different work-

load types and input sizes, and matches an actual job with a cluster under two conditions: (i) the CV of the cluster is smaller than a specified threshold (e.g. 0.5), which indicates the anonymous jobs in this cluster are closely similar to each other; (ii) the change in this CV is smaller than a threshold (e.g. 0.1) after the actual job is added to the cluster. This means the workload characteristics of the added job are sufficiently similar to those of the anonymous jobs in the cluster. If multiple matched actual jobs are found for one cluster, the combiner selects the job resulting the smallest CV change. Finally, the combiner produces workload replaying scripts as the output.

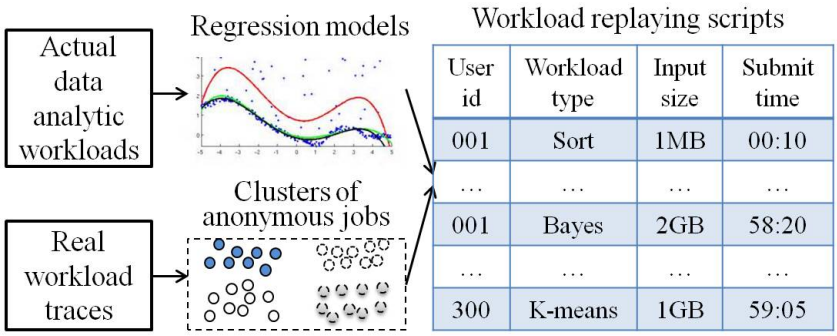


**Fig. 2.** The matching process of real and synthetic data analysis jobs

Note that BigDataBench-MT provides two ways of using the above combiner. First, it directly provides some workload replaying scripts, which are the combination results of representative actual workloads (e.g. Hadoop Sort and WordCount) and the Google workload trace. Second, it also supports benchmark users to directly use the above combination technique to match their own data analysis jobs with Google anonymous jobs.

### 3.2 Multi-tenant Workload Generator

Based on workload replaying scripts, the workload generator applies a multi-tenant mechanism to generate a mix of workloads using two steps. First, the generator extracts the tenant information from the scripts. For the service and data analysis workloads, this tenant information represents the number of concurrent end users and submitters of analytic jobs, respectively. Second, the generator creates a client for each tenant and emulates the scenarios that a number of end users/job submitters concurrently submit requests/jobs to the system. This multi-tenant framework allows the flexible adjustment of workload scales with guarantee of their realistic mixes. For example, benchmark users can double or halve the size of concurrent tenants, after which the distributions of requests/jobs submitted by these tenants still correspond to those in real workload traces.

# 4 Demonstration Description

## 4.1 Chosen Workloads and Workload Traces

In our demonstration, benchmark users want to evaluate their data center systems using a mix of service and data analysis workloads. The Nutch web search engine [5] is used as the example service workload and four Hadoop workloads are used as the example data analysis workloads. The chosen Hadoop workloads have a variety of workload characteristics: WordCount and Naïve Bayes classification are typical CPU-intensive workloads with integer and float point calculations; Sort is the typical I/O-intensive workload and PageIndex is the workload having similar demands for CPU and I/O resources. Both the data generators [42] and workloads in the demo can be obtained from BigDataBench [8].

Our demo uses a 24-hour user query logs from Sogou, which include 1,724,264 queries from 519,876 end users, as the basis to generate realistic search engine service; and uses a 24-hour cluster workload trace from Google, which includes 37,842 anonymous jobs from 2,261 job submitters, as the basis to generate realistic Hadoop jobs.

## 4.2 System Demonstration

BigDataBench-MT provides a visual interface in the *Benchmark User Portal* to help benchmark users make appropriate benchmarking decisions. This portal provides users necessary information, allows them input benchmarking requirements and executes system evaluations on their behalf. The whole process consists of three steps, as shown in Figures 3, 4 and 5, respectively.

*Step 1. Specification of tested machines and workloads.* The first step of the demo presents an overview of workload traces (i.e. Sogou and Google traces) and the data center status, including the six types of machines, their machine number and configurations, and the user, job and task statistics in these machines. This information assists benchmark users to select the type and number of machines to be evaluated, and the workloads they want to use. Suppose users select *Type Four* of the machines with 2 process cores and 4GB memory and 100 machines to be tested, the workload traces belonging to these machines are extracted and forwarded to the next step.

*Step 2. Selection of benchmarking period and scale.* At this step, users have the option to select the period and scale for their benchmarking scenarios. To facilitate this selection, BigDataBench-MT shows the statistic information of both the service workload (including its number of requests and end users per second) and the data analysis workloads (including their number of jobs and average CPU, memory resource usages) at each of the 24 hours. Suppose users select the benchmarking period of 12:00 to 13:00 and the scale factor is 1 (that is, no scaling is needed). The workload traces belonging to this period are selected for step 3.


**Fig. 3.** System Demonstration Screenshots: Step 1


**Fig. 4.** System Demonstration Screenshots: Step 2

*Step 3. Generation of mixed workloads.* After both workloads and traces have been selected, the final step employs the combiner described in Section 3.1 to generate workload replaying scripts for both the service and data analysis workloads, and sends these scripts as feedback to users. In the matching of actual Hadoop jobs with anonymous ones, we tested each Hadoop workload type using 20 different input sizes to build the regression models. Based on the replaying scripts, benchmark users can press the "Generate mixed workload" button to trigger the multi-tenant workload generator, in which each tenant is an independent workload generator and multiple tenants generate a mix of realistic workloads.

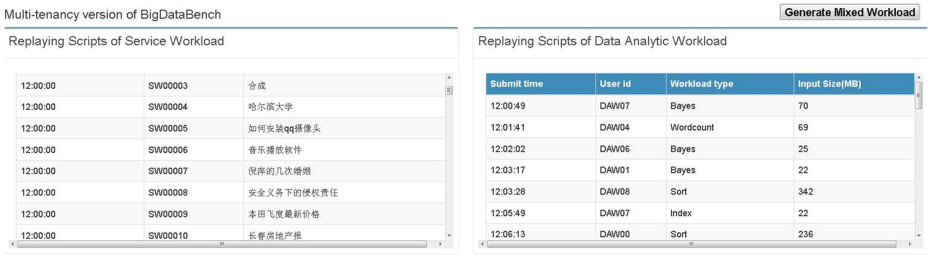


**Fig. 5.** System Demonstration Screenshots: Step 3

## 5 Future Work

There are multiple avenues for extending the functionality of our benchmark tool. A first step will be to support more actual workloads. Given that there are 33 actual workloads in the BigDataBench and many workloads (e.g. Bayes classification and WordCount) have three versions of implementations (Hadoop, Spark and MPI), adding more workloads to BigDataBench-MT will be helpful to support wider benchmarking scenarios. We also plan to extend our multi-tenant workload generator to support different classes of tenants and allow users to apply different priority disciplines in workload generation.

## 6 Acknowledgements

This work is supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015308), the National Natural Science Foundation of China (Grant No.61502451), and the Key Technology Research and Development Programs of Guangdong Province, China (Grant No.2015B010108006).

## References


1. Apache hbase. http://hbase.apache.org/.
2. Apache hive. https://cwiki.apache.org/confluence/display/Hive/Home.
3. Apache impala. http://impala.io/.
4. Apache mahout. http://mahout.apache.org/.
5. Apache nutch. http://nutch.apache.org/.
6. Apache spark. https://spark.apache.org/.
7. Big data benchmark by amplab of uc berkeley. https://amplab.cs.berkeley.edu/benchmark/.
8. Bigdatabench. http://prof.ict.ac.cn/BigDataBench/.
9. Grep. http://wiki.apache.org/hadoop/Grep.
10. Gridmix. https://hadoop.apache.org/docs/r1.2.1/gridmix.html.
11. Hadoop ecosystems. https://hadoopecosystemtable.github.io/.
12. Mysql database. https://www.mysql.com/.
13. Nnbench. http://grepcode.com/file/repo1.maven.org/maven2/org.apache.hadoop/hadoop-mapred-test/0.22.0/org/apache/hadoop/hdfs/NNBench.java/.
14. Oprofile. http://oprofile.sourceforge.net/.
15. Oracle database. http://www.oracle.com/.
16. Perf. https://perf.wiki.kernel.org/.
17. Pigmix. https://cwiki.apache.org/confluence/display/PIG/PigMix.
18. Sogou user query logs. http://www.sogou.com/labs/dl/q-e.html.
19. Sort. http://wiki.apache.org/hadoop/Sort.
20. Specweb99 benchmark. https://www.spec.org/web2009/.
21. Swim. https://github.com/SWIMProjectUCB/SWIM/wiki.
22. Terasort. https://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html.
23. Testdfsio. https://support.pivotal.io/hc/en-us/articles/200864057-Running-DFSIO-mapreduce-benchmark-test/.
24. Tpc benchmarks. http://www.tpc.org/.
25. Tpc-w benchmark. http://www.tpc.org/tpcw/.
26. Wordcount. http://wiki.apache.org/hadoop/WordCount.
27. T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. Linkbench: a database benchmark based on the facebook social graph. In *SIGMOD'13*, pages 1185–1196. ACM, 2013.
28. S. Barahmand and S. Ghandeharizadeh. Bg: A benchmark to evaluate interactive social networking actions. In *CIDR'13*. Citeseer, 2013.
29. M. Capotă, T. Hegeman, A. Iosup, A. Prat-Pérez, O. Erling, and P. Boncz. Graphalytics: A big data benchmark for graph-processing platforms. In *Proceedings of the GRADES'15*, page 7. ACM, 2015.
30. Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *VLDB'12*, 5(12):1802–1813, 2012.
31. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *SoCC'10*, pages 143–154. ACM, 2010.
32. J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.
33. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 205–220. ACM, 2007.

34. M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the clouds: A study of emerging workloads on modern hardware. Technical report, 2011.
35. A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In *SIGMOD'13*, pages 1197–1208. ACM, 2013.
36. R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98, 2014.
37. R. Han, J. Wang, F. Ge, J. L. Vazquez-Poletti, and J. Zhan. Sarp: producing approximate results with small correctness losses for cloud interactive services. In *CF'15*, page 22. ACM, 2015.
38. R. Han, J. Wang, S. Huang, C. Shao, S. Zhan, J. Zhan, and J. L. Vazquez-Poletti. Sarp: producing approximate results with small correctness losses for cloud interactive services. In *ICPP'15*, pages 490–499. IEEE, 2015.
39. S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW'13)*, pages 41–51. IEEE, 2010.
40. K. Kim, K. Jeon, H. Han, S.-g. Kim, H. Jung, and H. Y. Yeom. Mrbench: A benchmark for mapreduce framework. In *ICPADS'08*, pages 11–18. IEEE, 2008.
41. A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.
42. Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, and J. Zhan. Bdgs: A scalable big data generator suite in big data benchmarking. In *Advancing Big Data Benchmarks*, pages 138–154. Springer, 2014.
43. R. Nambiar. A standard for benchmarking big data systems. In *BigData'14*, pages 18–20. IEEE, 2014.
44. A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis. In *SIGMOD'09*, pages 165–178. ACM, 2009.
45. D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML'00*, pages 727–734, 2000.
46. C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM, 2012.
47. V. Saletore, K. Krishnan, V. Viswanathan, M. E. Tolentino, et al. Hcbench: Methodology, development, and characterization of a customer usage representative big data/hadoop benchmark. In *IISWC'13*, pages 77–86. IEEE, 2013.
48. A. Sangroya, D. Serrano, and S. Bouchenak. Mrbs: towards dependability benchmarking for hadoop mapreduce. In *EuroPar'12*, pages 3–12. Springer, 2013.
49. D. Shankar, X. Lu, M. Wasi-ur Rahman, N. Islam, and D. K. D. Panda. A micro-benchmark suite for evaluating hadoop mapreduce on high-performance networks. In *BPOE'14*, pages 19–33. Springer, 2014.
50. R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah. Serving large-scale batch computed data with project voldemort. In *FAST'12*, pages 18–18. USENIX Association, 2012.
51. L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, et al. Bigdatabench: a big data benchmark suite from internet services. In *HPCA'14*, pages 488–499. IEEE, 2014.