Benchmarking Data Lakes Featuring Structured and Unstructured Data with DLBench

Pegdwendé N. Sawadogo[1], Jérôme Darmont[1], and Camille Noûs[2]

[1] Université de Lyon, Lyon 2, UR ERIC
5 avenue Pierre Mendès France, F69676 Bron Cedex, France
{pegdwende.sawadogo,jerome.darmont}@univ-lyon2.fr
[2] Université de Lyon, Lyon 2, Laboratoire Cogitamus
camille.nous@cogitamus.fr

Abstract. In the last few years, the concept of data lake has become trendy for data storage and analysis. Thus, several approaches have been proposed to build data lake systems. However, these proposals are difficult to evaluate as there are no commonly shared criteria for comparing data lake systems. Thus, we introduce DLBench, a benchmark to evaluate and compare data lake implementations that support textual and/or tabular contents. More concretely, we propose a data model made of both textual and CSV documents, a workload model composed of a set of various tasks, as well as a set of performance-based metrics, all relevant to the context of data lakes. As a proof of concept, we use DLBench to evaluate an open source data lake system we previously developed.

Keywords: Data lakes · Benchmarking · Textual Documents · Tabular data

1 Introduction

Over the last decade, the concept of data lake has emerged as a reference for data storage and exploitation. A data lake is a large repository for storing and analyzing data of any type and size, kept in their raw format [3]. Data access and analyses from data lakes largely rely on metadata [12], making data lakes flexible enough to support a broader range of analyses than traditional data warehouses. Data lakes are thus handy for both data retrieval and data content analysis.

However, the concept of data lake still lacks standards [15]. Thus, there is no commonly shared approach to build, nor to evaluate a data lake. Moreover, existing data lake architectures are often evaluated in diverse and specific ways, and are hardly comparable with each other. Therefore, there is a need of benchmarks to allow objective and comparative evaluation of data lake implementations. There are several benchmarks for big data systems in the literature, but none of them considers the wide range of possible analyses in data lakes.

Hence, we propose in this paper the Data Lake Benchmark (DLBench) to evaluate data management performance in data lake systems. We particularly focus in this first instance on textual and tabular contents, which are often included in data lakes. DLBench is data-centric, i.e., it focuses on a data management objective, regardless of the underlying technologies [2]. We also designed it with Gray's criteria for a "good" benchmark in mind, namely relevance, portability, simplicity and scalability [8].

More concretely, DLBench features a data model that generates textual and tabular documents. By tabular documents, we mean spreadsheet or Comma Separated Value (CSV) files whose integration and querying is a common issue in data lakes. A scale factor parameter SF allows to vary data size in predetermined proportions. DLBench also features a workload model, i.e., a set of analytical operations relevant to the context of data lakes with textual and/or tabular content. Finally, we propose a set of performance-based metrics to evaluate such data lake implementations, as well as an execution protocol to execute the workload model on the data model and compute the metrics.

The remainder of this paper is organized as follows. In Section 2, we show how DLBench differs from existing benchmarks. In Section 3, we provide DLBench's full specifications. In Section 4, we exemplify how DLBench works and the insights it provides. Finally, in Section 5, we conclude this paper and present research perspectives.

2 Related Works

Benchmarking data lakes mainly relates to two benchmark categories, namely big data and text benchmarks. In this section, we present recent works in these categories and discuss their limitations with respect to our benchmarking objectives.

2.1 Big Data Benchmarks

Big data systems are so diverse that each of big data benchmarks in the literature only target a part of big data requirements [1]. The *de facto* standard TPC-H [18] and TPC-DS [20] issued by the Transaction Processing Performance Council are still widely used to benchmark traditional business intelligence systems. They provide data models that reflect a typical data warehouse, as well as a set of typical business queries, mostly in SQL. BigBench [7] is another reference benchmark that addresses SQL querying on data warehouses. In addition, BigBench adaptations [6, 10] include more complex big data analysis tasks, namely sentiment analysis over short texts.

TPCx-HS [19] is a quite different benchmark that aims to evaluate systems running on Apache Hadoop[3] or Spark[4]. For this purpose, only a sorting workload helps measuring performances. HiBench [9] also evaluates Hadoop/Spark systems, but with a broader range of workloads, i.e., ten workloads including SQL aggregations and joins, classification, clustering and sorts [11]. In the same

[3] https://hadoop.apache.org/
[4] http://spark.apache.org/

line, TPCx-AI [21], which is still in development, includes more analysis tasks relevant to big data systems, such as advanced machine learning tasks for fraud detection and product rating.

2.2 Textual Benchmarks

In this category, we consider big data benchmarks with a consequent part of text on one hand, and purely textual benchmarks on the other hand. BigDataBench [23] is a good representative of the first category. It indeed includes a textual dataset made of Wikipedia[5] pages, as well as classical information retrieval workloads such as *Sort*, *Grep* and *Wordcount* operations.

One of the latest purely textual benchmarks is TextBenDS [22], which aims to evaluate performances of text analysis and processing systems. For this purpose, TextBenDS proposes a tweet-based data model and two types of workloads, namely *Top-K keywords* and *Top-K documents* operations. Other purely textual benchmarks focus on language analysis tasks, e.g., Chinese [25] and Portuguese [5] text recognition, respectively.

2.3 Discussion

None of the aforementioned benchmarks proposes a workload sufficiently extensive to reflect all relevant operations in data lakes. In the case of structured data, most benchmark workloads only consider SQL operations (TPC-H, TPC-DS, HiBench). More sophisticated machine learning operations remain marginal, while they are common analyses in data lakes. Moreover, the task of finding related data (e.g., joinable tables) is purely missing, while it is a key feature of data lakes.

Existing textual workloads are also insufficient. Admittedly, BigDataBench's *Grep* and TextBenDS's *Top-K documents* operation are relevant for data search. Similarly, *Top-K keywords* and *WordCount* are relevant to assess documents aggregation [9, 22]. However, other operations such as finding most similar documents or clustering documents should also be considered.

Thus, our DLBench benchmark stands out, with a broader workload that features both data retrieval and data content analysis operations. DLBench's data model also differs from most big data benchmarks as it provides raw tabular files, inducing an additional data integration challenge. Moreover, DLBench includes a set of long textual documents that induces a different challenge than short texts such as tweets [22] and Wikipedia articles [23]. Finally, DLBench is data-centric, unlike big data benchmarks that focus on a particular technology, e.g., TPCx-HS and HiBench.

[5] https://en.wikipedia.org/

3 Specification of DLBench

In this section, we first provide a thorough description of DLBench's data and workload model. Then, we propose a set of metrics and introduce an assessment protocol to evaluate and/or compare systems using DLBench.

3.1 Data Model

Data Description DLBench includes two types of data to simulate a data lake: textual documents and tabular data. Textual documents are scientific articles that span from few to tens of pages. They are written in French and English and their number amounts to 50,000. Their overall volume is about 62 GB.

Tabular data are synthetically derived from a few CSV files containing Canadian government open data. Although such data are often considered as structured, they still need integration to be queried and analyzed effectively. DLBench features up to 5,000 tabular files amounting to about 1,4 GB of data.

The amount of data in the benchmark can be customised through scale factor parameter SF, which is particularly useful to measure a system's performance when data volume increases. SF ranges from 1 to 5. Table 1 describes the actual amount of data obtained with values of SF.

Table 1: Amount of data per SF value

Scale factors	$SF = 1$	$SF = 2$	$SF = 3$	$SF = 4$	$SF = 5$
Nb. of textual documents	10,000	20,000	30,000	40,000	50,000
Nb. of tabular files	1,000	2,000	3,000	4,000	5,000
Textual documents' size (GB)	8.0	24.9	37.2	49.6	62.7
Tabular files' size (GB)	0.3	0.6	0.8	1.1	1.4

DLBench's data come with metadata catalogues that can serve in data integration. More concretely, we generate from textual documents catalogue information on *year*, *language* and *domain* (discipline) to which each document belongs. Similarly, we associate in the tabular file catalogue a *year* with each file. This way, we can separately query each type of data through its specific metadata. We can also jointly query textual documents and tabular files through the *year* field.

Eventually, textual documents are generated independently from tabular files. Therefore, each type of data can be used apart from the other. In other words, DLBench can be used to assess a system that contains either textual documents only, tabular files only, or both. When not using both types of data, the workload model must be limited to its relevant part.

Data Extraction We extract textual data from HAL[6], a French open data repository dedicated to scientific document diffusion. We opted for scientific documents as most are long enough to provide complexity and reflect most actual use cases in textual data integration systems, in contrast with shorter documents such as reviews and tweets.

Although HAL's access is open, we are not allowed to redistribute data extracted from HAL. Thus, we provide instead a script that extracts a user-defined amount of documents. This script and a usage guide are available online for reuse[7]. Amongst all available documents in HAL, we restrict to scientific articles whose length is homogeneous, which amounts to 50,000 documents. While extracting documents, the script also generates the metadata catalogue described above.

Tabular data are reused from an existing benchmark [13]. These are actually a set of 5,000 synthetic tabular data files generated from an open dataset stored inside a SQLite[8] database. Many of the columns in the tables contain similar data and can therefore be linked, making this dataset suitable to assess structured data integration as performed in data lakes.

We apply on this original dataset[9] a script to extract all (or a part) of the tables in the form of raw CSV files. As for textual documents, this second script also generates a metadata catalogue. The script as well as guidelines are available online[7].

3.2 Workload Model

To assess and compare data lakes across different implementations and systems, some relevant tasks are needed. Thus we specify in this section instances of probable tasks in textual and tabular data integration systems. Furthermore, we translate each task into concrete, executable queries (Table 2).

Data Retrieval Tasks are operations that find data bearing given characteristics. Three main ways are usually exploited to retrieve data in a lake. They are relevant for both tabular data and textual documents. However, we mainly focus on data retrieval from textual documents, as they represent the largest amount of data.

1. *Category filters* consist in filtering data using tags or data properties from the metadata catalogue. In other words, it can be viewed as a navigation task.
2. *Term-based search* pertains to find data, with the help of an index, from all data files that contain a set of keywords. Keyword search is especially relevant for textual documents, but may also serve to retrieve tabular data.

[6] https://hal.archives-ouvertes.fr/
[7] https://github.com/Pegdwende44/DLBench
[8] https://www.sqlite.org/
[9] https://storage.googleapis.com/table-union-benchmark/large/benchmark.sqlite

3. *Related data search* aims to, from a specified data file, retrieve similar data. It can be based on, e.g., column similarities in tabular data, or semantic similarity between textual documents.

Textual Document Analysis/Aggregation tasks work on data contents. Although textual documents and tabular data can be explored with the same methods, they require more specific techniques to be jointly analyzed or aggregated in data lakes.

4. *Document scoring* is a classical information retrieval task that consists in providing a score for each document with respect to how it matches a set of terms. Such scores can be calculated by diverse ways, e.g., with the Elastic-Search [4] scoring algorithm. In all cases, scores depend on the appearance frequency of query terms in the document to score and in the corpus, and also the document's length. This operation is actually very similar to computing top-k documents.
5. *Document highlights* extract a concordance from a corpus. A concordance is a list of snippets where a set of terms appear. It is also a classical information retrieval task that provides a sort of summary of documents.
6. *Document top keywords* are another classical way to summarize and aggregate documents [14]. Computing top keywords is thus a suitable task to assess systems handling textual documents.
7. *Document text mining.* In most data lake systems, data are organized in collections, using tags for example. Here, we propose a data mining task that consists either in representing each collection of documents with respect to the others, or in grouping together similar collections with respect to their intrinsic vocabularies. In the first case, we propose a Principal Component Analysis (PCA) [24] where statistical individuals are document collections. PCA could, for example, out put an average bag of words for each collection. In the second case, we propose a KMeans [17] clustering to detect groups of similar collections.

Tabular Data Analysis/Queries Finally, we propose specific tasks suitable for integrated tabular files.

8. *Simple table queries.* We first propose to evaluate a data lake system's capacity to answer simple table queries through a query language such as SQL. As we are in a context of raw tabular data, language-based querying is indeed an important challenge to address.
9. *Complex table queries.* In line with the previous task, we propose to measure how the system supports advanced queries, namely join and grouping queries.
10. *Tuple mining.* An interesting way to analyze tabular data is either to represent each row with respect to the others or to group together very similar rows. We essentially propose here the same operation as Task #7 above, except that statistical individuals are table rows instead of textual documents. To achieve such an analysis, we only consider numeric values.

Table 2: Query instances

Task	**Query**	
		Data retrieval
#1	Q1a	Retrieve documents written in *French*
	Q1b	Retrieve documents written in *English* and edited in *December*
	Q1c	Retrieve documents whose domains are *math* or *info*, written in *English* and edited in *2010*, *2012* or *2014*
#2	Q2a	Retrieve data files (documents or tables) containing the term *university*
	Q2b	Retrieve data files containing the terms *university*, *science* or *research*
#3	Q3a	Retrieve the top 5 documents similar to any given document
	Q3b	Retrieve 5 tables joinable to table *t_dc9442ed0b52d69c____c11_1____1*
		Textual Document Analysis/Aggregation
#4	Q4a	Calculate documents scores w.r.t. the terms *university* and *science*
	Q4b	Calculate documents scores w.r.t. the terms *university*, *research*, *new* and *solution*
#5	Q5a	Retrieve documents concordance w.r.t. the terms *university* and *science*
	Q5b	Retrieve documents concordance w.r.t. the terms *university*, *science* *new* and *solution*
#6	Q6a	Find top 10 keywords from all documents (stopwords excluded)
#7	Q7a	Run a PCA with documents merged by *domains*
	Q7b	Run a 3-cluster KMeans clustering with documents merged by *domains*
		Tabular Data Analysis/Queries
#8	Q8a	Retrieve all tuples from table *t_e9efd5cda78af711____c11_1____1*
	Q8b	Retrieve tuples from table *t_e9efd5cda78af711____c11_1____1* whose column *PROVINCE* bears the value *BC*
#9	Q9a	Calculate the average of columns *Unnamed: 12*, *13*, and *20* from table *t_356fc1eaad97f93b____c15_1____1* grouped by *Unnamed: 2*
	Q9b	Run a left join query between tables *PED_SK_DTL_SNF____c7_0____1* and *t_285b3bcd52ec0c86____c13_1____1* w.r.t. columns named *SOILTYPE*
#10	Q10a	Run a PCA on the result of query *Q9a*
	Q10b	Run a 3-cluster KMeans clustering on the result of query *Q9a*

3.3 Performance Metrics

In this section, we propose a set of three metrics to compare and assess data lake implementations.

1. **Query execution time** aims to measure the time necessary to run each of the 20 query instances from Table 2 on the tested data lake architecture. This metric actually reports how efficient the lake's metadata system is, as it serves to integrate raw data, and thus make analyses easier and faster. In the case where certain queries are not supported, measures are only computed on the supported tasks.
2. **Metadata size** measures the amount of metadata generated by the system. It allows to balance the execution time with the resulting storage cost.

3. **Metadata generation time** encompasses the generation of all the lake's metadata. This also serves to balance query execution time.

We did not include other possible metrics such as actually used RAM and CPU because they are hard to measure. However, we recommend interpreting benchmark results while taking into account available RAM and CPU.

3.4 Assessment Protocol

The three metrics from Section 3.3 are measured through an iterative process for each scale factor $SF \in \{1, 2, 3, 4, 5\}$. Each iteration consists in four steps (Algorithm 1).

1. **Data generation** is achieved with the scripts specified in Section 3.1[7].
2. **Data integration.** Raw data now need to be integrated in the data lake system through the generation and organization of metadata. This step is specific to each system, as there are plethora of ways to integrate data in a lake.
3. **Metadata size and generation time computing** consists in measuring metrics the total size of generated metadata and the time taken to generate all metadata, with respect to the current SF.
4. **Query execution time computation** involves computing the running time of each individual query. To mitigate any perturbation, we average the time of 10 runs for each query instance. Let us notice that all timed executions must be warm runs, i.e., each of the 20 query instances must first be executed once (a cold run not taken into account in the results).

Algorithm 1: Assessment protocol

```
Result: metric_1, metric_2, metric_3
metric_1 ← [ ][ ]; metric_2 ← [ ]; metric_3 ← [ ];
for SF ← 1 to 5 do
    generate_benchmark_data(SF);
    generate_and_organize_metadata(SF);
    metric_2[SF] ← retrieve_metadata_generation_time(SF);
    metric_3[SF] ← retrieve_metadata_size(SF);
    for i ← 1 to 20 do
        run_query(i, SF);
        response_times ← [ ];
        for j ← 1 to 10 do
            response_times[j] ← run_query(i, SF);
        end
        metric_1[SF][i] ← average(response_times);
    end
end
```

4 Proof of Concept

4.1 Overview of AUDAL

To demonstrate the use of DLBench, we evaluate AUDAL [16], a data lake system designed as part of a management science project, to allow automatic and advanced analyses on various textual documents (annual reports, press releases, websites, social media posts...) and spreadsheet files (information about companies, stock market quotations...). The AUDAL system uses an extensive metadata system stored inside MongoDB[10], Neo4J[11], SQLite[8] and ElasticSearch[12] to support numerous analyses.

AUDAL provides ready-to-use analyses via a representational state transfer application programming interface (REST API) dedicated to data scientists, and also through a Web-based analysis platform designed for business users.

4.2 Setup and Results

AUDAL is implemented on a cluster of three VMware virtual machines (VMs). The first VM has a 7-core Intel-Xeon 2.20 GHz processor and 24 GB of RAM. It runs the API and also supports metadata extraction. Both other VMs have a mono-core Intel-Xeon 2.20 GHz processor and 24 GB of RAM. Each of the three VMs hosts a Neo4J instance, an ElasticSearch instance and a MongoDB instance to store AUDAL's metadata.

The results achieved with DLBench show that AUDAL scales quite well (Figures 1-6). Almost all task response times are indeed either constant (Tasks #3, #7, #8, #9 and #10) or grow linearly with SF (Tasks #1, #2, and #4 to #6). In addition, we observe that except Task #6 (that takes up to 92 seconds), all execution times are reasonable considering the modest capabilities of our hardware setup. Eventually, metadata generation time and size scale linearly and almost-linearly, respectively (Figure 7). We can also see that metadata amount to about half the volume of raw data, which illustrates how extensive AUDAL's metadata are. We observe some fluctuations in the results, with sometimes negative slopes while SF increases. Such variations are due to external, random factors such as network load or the Java garbage collector starting running. However, the influence on the runtime is negligible (of the order of a tenth of a second) and is only visible on simple queries that run in half a second.

5 Conclusion

In this paper, we introduce DLBench, a benchmark for data lakes with textual and/or tabular contents. To the best of our knowledge, DLBench is the first data lake benchmark. DLBench features: 1) a data model made of a corpus

[10] https://www.mongodb.com
[11] https://neo4j.com
[12] https://www.elastic.co

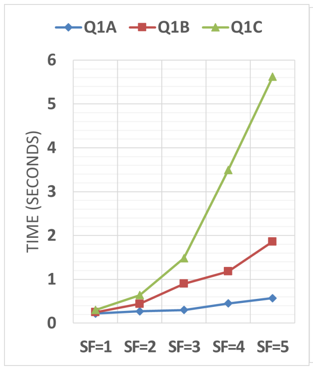

Fig. 1: Task #1 response times

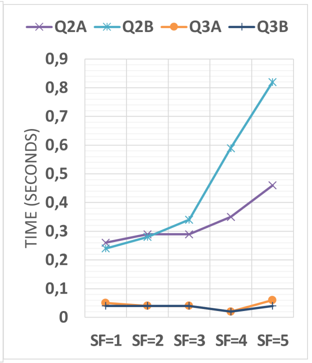

Fig. 2: Tasks #2 & #3 response times

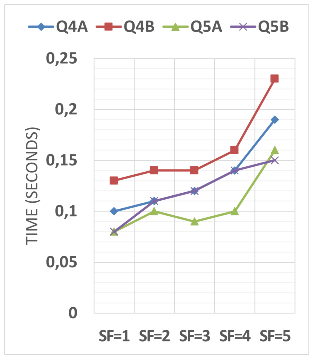

Fig. 3: Tasks #4 & #5 response times

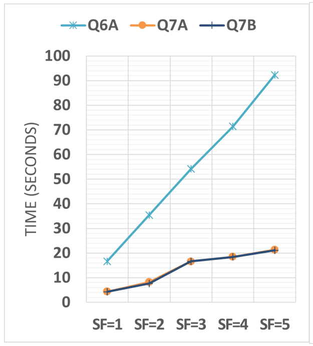

Fig. 4: Tasks #6 & #7 response times

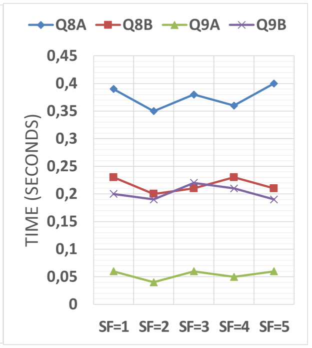

Fig. 5: Tasks #8 & #9 response times

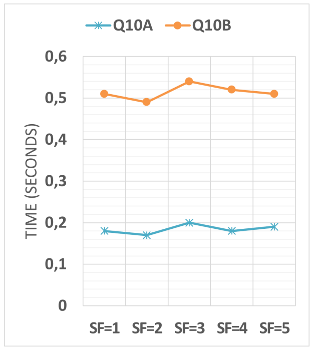

Fig. 6: Task #10 response times

of long, textual documents on one hand, and a set of raw tabular data on the other hand; 2) a query model of twenty query instances across ten different tasks; 3) three relevant metrics to assess and compare data lake implementations; and 4) an execution protocol. Finally, we demonstrate the use of DLBench by assessing the AUDAL data lake [16], highlighting that the AUDAL system scales quite well, especially for data retrieval and tabular data querying.

Future works include an extension of the structured part of DLBench's data model with an alternative, larger dataset. Another enhancement of DLBench could consists in providing an overview of value distributions in generated data. Finally, we plan to perform a comparative study of existing data lake systems using DLBench.

Acknowledgments

P.N. Sawadogo's PhD is funded by the Auvergne-Rhône-Alpes Region through the AURA-PMI project.

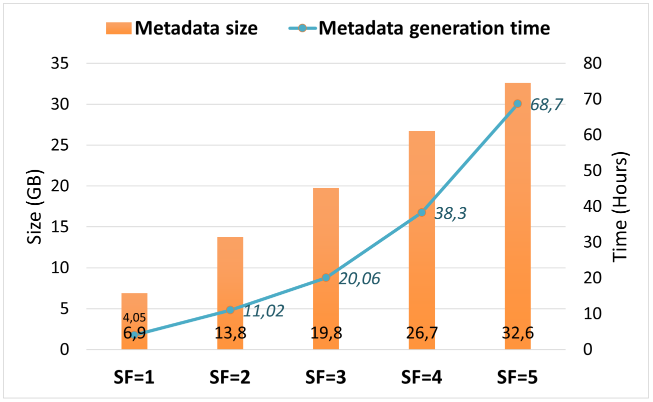

Fig. 7: Metadata size and generation time

References

1. Bajaber, F., Sakr, S., Batarfi, O., Altalhi, A.H., Barnawi, A.: Benchmarking big data systems: A survey. Comput. Commun. **149**, 241–251 (2020). https://doi.org/10.1016/j.comcom.2019.10.002
2. Darmont, J.: Data-centric benchmarking. In: Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics, pp. 342–353. IGI Global (2019)
3. Dixon, J.: Pentaho, Hadoop, and Data Lakes (October 2010), https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/
4. Elasticsearch: Theory Behind Relevance Scoring (Sptember 2019), https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html
5. Fialho, P., Coheur, L., Quaresma, P.: Benchmarking natural language inference and semantic textual similarity for portuguese. Inf. **11**(10), 484 (2020). https://doi.org/10.3390/info11100484
6. Ghazal, A., Ivanov, T., Kostamaa, P., Crolotte, A., Voong, R., Al-Kateb, M., Ghazal, W., Zicari, R.V.: Bigbench V2: the new and improved bigbench. In: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017. pp. 1225–1236. IEEE Computer Society (2017). https://doi.org/10.1109/ICDE.2017.167
7. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., Jacobsen, H.: Bigbench: towards an industry standard benchmark for big data analytics. In: Ross, K.A., Srivastava, D., Papadias, D. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013. pp. 1197–1208. ACM (2013). https://doi.org/10.1145/2463676.2463712, https://doi.org/10.1145/2463676.2463712
8. Gray, J.: Database and transaction processing performance handbook. (1993), http://jimgray.azurewebsites.net/benchmarkhandbook/chapter1.pdf
9. 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 (2010). https://doi.org/10.1109/ICDEW.2010.5452747

10. Ivanov, T., Ghazal, A., Crolotte, A., Kostamaa, P., Ghazal, Y.: Corebigbench: Benchmarking big data core operations. In: Tözün, P., Böhm, A. (eds.) Proceedings of the 8th International Workshop on Testing Database Systems, DBTest@SIGMOD 2020, Portland, Oregon, June 19, 2020. pp. 4:1–4:6. ACM (2020). https://doi.org/10.1145/3395032.3395324
11. Ivanov, T., Rabl, T., Poess, M., Queralt, A., Poelman, J., Poggi, N., Buell, J.: Big Data Benchmark Compendium. In: Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things - 7th TPC Technology Conference, TPCTC 2015, Kohala Coast, HI, USA. pp. 135–155 (September 2015). https://doi.org/10.1007/978-3-319-31409-9_9
12. Maccioni, A., Torlone, R.: KAYAK: A Framework for Just-in-Time Data Preparation in a Data Lake. In: International Conference on Advanced Information Systems Engineering (CAiSE 2018), Tallin, Estonia. pp. 474–489 (june 2018). https://doi.org/10.1007/$978 - 3 - 319 - 91563 - 0_2 9$
13. Nargesian, F., Zhu, E., Pu, K.Q., Miller, R.J.: Table Union Search on Open Data. Proceedings of the VLDB Endowment **11**, 813–825 (March 2018). https://doi.org/10.14778/3192965.3192973
14. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Top-keyword: An Aggregation Function for Textual Document OLAP. In: 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008), Turin, Italy. pp. 55–64 (September 2008). https://doi.org/10.1007/$978 - 3 - 540 - 85836 - 2_6$
15. Russom, P.: Data Lakes Purposes, Practices, Patterns, and Platforms. TDWI research (2017)
16. Scholly, E., Sawadogo, P.N., Liu, P., Espinosa-Oviedo, J.A., Favre, C., Loudcher, S., Darmont, J., Noûs, C.: Coining goldMEDAL: A New Contribution to Data Lake Generic Metadata Modeling. In: 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP@EDBT/ICDT 2021), Nicosia, Cyprus. pp. 31–40 (March 2021)
17. Steinley, D.: K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology **59**(1), 1–34 (2006)
18. Transaction Processing Performance Council: TPC Benchmark H - Standard Specification (version 2.18.0). http://www.tpc.org/tpch/ (2014)
19. Transaction Processing Performance Council: TPC Express Benchmark HS - Standard Specification (version 2.0.3). http://www.tpc.org/tpcds/ (2018)
20. Transaction Processing Performance Council: TPC Benchmark DS - Standard Specification (version 2.13.0). http://www.tpc.org/tpcds/ (2020)
21. Transaction Processing Performance Council: TPC Express AI - Draft Specification (version 0.6). http://tpc.org/tpcx-ai/default5.asp (2020)
22. Truica, C., Apostol, E.S., Darmont, J., Assent, I.: Textbends: a generic textual data benchmark for distributed systems. Inf. Syst. Frontiers **23**(1), 81–100 (2021). https://doi.org/10.1007/s10796-020-09999-y
23. 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 (2014)
24. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and Intelligent Laboratory Systems **2**(1), 37–52 (1987). https://doi.org/10.1016/0169-7439(87)80084-9
25. Zhu, Y., Xie, Z., Jin, L., Chen, X., Huang, Y., Zhang, M.: SCUT-EPT: new dataset and benchmark for offline chinese text recognition in examination paper. IEEE Access **7**, 370–382 (2019). https://doi.org/10.1109/ACCESS.2018.2885398