DLBench+: a Benchmark for Quantitative and Qualitative Data Lake Assessment

Pegdwendé Nicolas Sawadogo, Jérôme Darmont

DLBench+: a Benchmark for Quantitative and Qualitative Data Lake Assessment

Pegdwendé N. Sawadogo and Jérôme Darmont

Université de Lyon, Lyon 2, UR ERIC
5 avenue Pierre Mendès France, F69676 Bron Cedex, France
{pegdwende.sawadogo,jerome.darmont}@univ-lyon2.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, such proposals are difficult to evaluate as there are no commonly shared criteria for comparing data lake systems. Thus, we introduce in this paper 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. Beyond a purely quantitative assessment, we also propose a methodology to qualitatively evaluate data lake systems through the assessment of user experience. As a proof of concept, we use DLBench+ to evaluate an open source data lake system we developed.

Keywords: Data lakes · Benchmarking · Textual Documents · Tabular data · Quality and metrics

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 [7]. Data access and analytics from data lakes largely rely on metadata [23], 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 [30]. 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 an objective and comparative evaluation of data lake implementations. Although there are several benchmarks for big data systems in the literature, none of them considers the wide range of analyses that are possible in data lakes.

To fill in this gap, we introduced the Data Lake Benchmark (DLBench) [31] to provide a quantitative performance evaluation of data lake systems managing textual and tabular contents. However, sheer quantitative performance is not enough. Another key issue is to make data accessible and valuable to users [28]. This is especially true for business users who are not expected to have sufficient skills to extract information from the lake themselves. Thus, there is also a need for qualitative performance evaluation related to user experience.

In this paper, we propose an enhanced version of DLBench, called DLBench+, which incorporates both quantitative and qualitative features to evaluate data lake systems in a more comprehensive way. DLBench+ remains technology-agnostic, i.e., it focuses on a data management objective, regardless of the underlying technologies [6]. The quantitative side of DLBench+ is designed with Gray's criteria for a "good" benchmark in mind, namely relevance, portability, simplicity and scalability [16]. The qualitative side of DLBench+ is based on the measure of user experience.

More concretely, DLBench+ features in its quantitative part a data model that generates textual and tabular documents, i.e., 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 quantitative metrics.

The qualitative part of DLBench+ features a set of metrics to measure the user experience and a survey-based protocol to measure them. The metrics we propose thus allow to evaluate the usability, the time-saving and the usefulness provided by the system from the users view. The measure of the usability is based on the System Usability Scale (SUS) methodology [5] which is widely used to evaluate software tools. The time-saving and usefulness measure are more specific to data lake systems.

Eventually, DLBench+ mostly focuses on the analysis capabilities of data lake systems. Evaluating the preprocessing phase would indeed presumably require a full benchmark on its own, i.e., a benchmark very similar to ETL (Extract, Transform, Load – even though data lakes usually adopt an ELT strategy) benchmarks that do already exist [33,42]. Yet, we still provide a simple way to evaluate preprocessing, by measuring preprocessing time (i.e., data integration and metadata extraction) and storage space overhead.

The remainder of this paper is organized as follows. In Section 2, we survey existing benchmarks and show how DLBench+ differs from them, as well as user experience evaluation. In Sections 3 and 4, we provide DLBench+'s specifications from the quantitative and qualitative point of view, respectively. In Section 5, we exemplify how DLBench+ works and the insights it provides by benchmarking the AUDAL data lake [32]. Finally, in Section 6, we conclude this paper and present research perspectives.

2 Related Works

Our benchmark proposal 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 [2]. The *de facto* standard TPC-H [35] and TPC-DS [37] issued by the Transaction Processing Performance Council (TPC) are still widely used to benchmark traditional business intelligence systems, i.e., data warehouses. They provide data models that reflect a typical data warehouse, as well as a set of typical business queries, mostly in SQL. BigBench [15] is another reference benchmark that addresses SQL querying on data warehouses. In addition, BigBench adaptations [14,18] include more complex big data analysis tasks, namely sentiment analysis over short texts.

TPCx-HS [36] is a quite different benchmark that aims to evaluate systems running on Apache Hadoop[1] or Spark[2]. For this purpose, only a sorting workload helps measuring performances. HiBench [17] 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 [19]. In the same line, TPCx-AI [38], 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 [40,12] is a good representative of the first category. It indeed includes a textual dataset made of Wikipedia[3] pages, as well as classical information retrieval workloads such as *Sort*, *Grep* and *Wordcount* operations. The PRIMEBALL benchmark [10] also belongs to this category with a workload and a dataset related to news articles management in cloud systems.

One of the latest purely textual benchmarks is TextBenDS [39], which aims to evaluate performance of text analysis and processing systems. For this purpose, TextBenDS proposes a tweet-based data model

[1] `https://hadoop.apache.org`
[2] `http://spark.apache.org`
[3] https://en.wikipedia.org

and two types of workloads, namely *Top-K keywords* and *Top-K documents* operations. The LSHTC benchmark [26] addresses more specifically the evaluation of text classification systems. It features a dataset including Web pages and a set of relevant metrics such as system accuracy, precision, recall and F_1 score. Similarly, Reagan et al.'s benchmark is dedicated to sentiment analysis tasks [29]. Eventually, other purely textual benchmarks focus on language analysis tasks, e.g., Chinese [43] and Portuguese [11] 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 [17,39]. However, other operations such as finding most similar documents or clustering documents should also be considered.

Thus, DLBench+ 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 [39] and Wikipedia articles [40]. Finally, DLBench+ is technology-agnostic, unlike big data benchmarks that focus on a particular technology, e.g., TPCx-HS and HiBench.

3 Quantitative Benchmarking

In this section, we first provide a thorough description of DLBench+'s quantitative part. We thus present the benchmark'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 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.

Figure 1 depicts DLBench+'s data and metadata as a constellation model, where *Document* and *Table* facts represent textual documents and tabular files, respectively. We organize metadata as dimensions. Both facts are all connected to the *Month* and *Term* dimensions, which allows simultaneously querying them. *Document* facts are also associated with the *Domain* and *Language* dimensions that reflect the field and writing language documents belongs to, respectively.

The amount of data in the benchmark can be customized through a 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.

DLBench+'s data come with metadata catalogs that can serve in data integration. More concretely, we generate from textual documents catalog information on *year*, *language* and *domain* (discipline) to which each document belongs. Similarly, we associate in the tabular file catalog 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

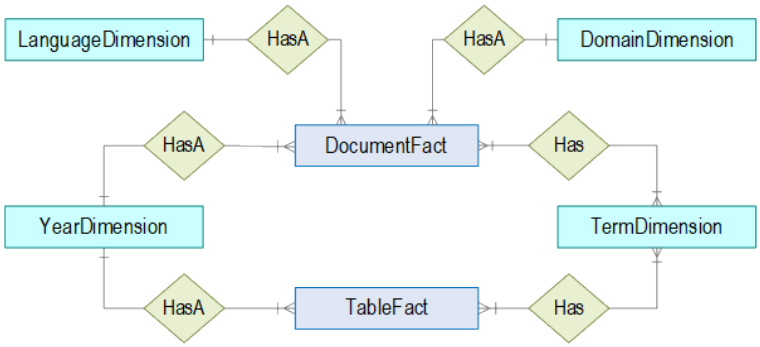

Fig. 1: DLBench+ data model

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

contains either textual documents only, tabular files only, or both. When not using both types of data, the workload model must be limited relevant queries.

Data Extraction We extract textual data from HAL[4], 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[5]. 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 catalog described above.

Tabular data are reused from an existing benchmark [25]. It is actually a set of 5,000 synthetic tabular data files generated from an open dataset stored inside a SQLite[6] 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[7] a script that extracts either all or a part of the tables, in the form of CSV files. As for textual documents, this second script also generates a metadata catalog. The script as well as guidelines are available online[5].

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).

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

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 catalog. 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.
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 ElasticSearch [8] scoring algorithm. In all cases, scores depend on the appearance frequency of query terms in the document to score and in the corpus, as well as 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 kind of document summary.
6. *Document top keywords* are another classical way to summarize and aggregate documents [27]. 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) [41] where statistical individuals are document collections. For example, PCA can output an average bag of words for each collection. In the second case, we propose a KMeans [34] 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.

3.3 Performance Metrics

In this section, we propose a set of three metrics to compare and assess data lake implementations. The choice of these metrics is motivated by the need to evaluate the performance of a data lake in terms of both analysis, on one hand, and data integration and metadata extraction, on the other hand.

Table 2: DLBench+ 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*
	Q2c	Retrieve data files containing terms like *universit**, *problem** or *research** (e.g., "*research**" must also match with the terms *researcher*, *researchers*, *researches*, etc.)
#3	Q3a	Retrieve the top 5 documents similar to any given document
	Q3b	Retrieve 5 tables joinable to table *t_dc*9442*ed*0*b*52*d*69*c*____*c*11_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*
	Q5c	Retrieve documents concordance w.r.t. the terms like *universit**, *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 tuples from table *t_e*9*efd*5*cda*78*af*711____*c*11_1____1
	Q8b	Retrieve tuples from table *t_e*9*efd*5*cda*78*af*711____*c*11_1____1 whose column *PROVINCE* bears the value *BC*
	Q8c	Count tuples from table *t_e*9*efd*5*cda*78*af*711____*c*11_1____1
#9	Q9a	Calculate the average of columns *Unnamed: 12*, *13*, and *20* from table *t_*356*fc*1*eaad*97*f*93*b*____*c*15_1____1 grouped by *Unnamed: 2*
	Q9b	Run a left join query between tables *PED_SK_DTL_SNF*____*c*7_0____1 and *t_*285*b*3*bcd*52*ec*0*c*86____*c*13_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*

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.

Eventually, we did not include other possible metrics, such as RAM and CPU usage. When benchmarking sufficiently big data, I/Os are preponderant *vs.* RAM and CPU. This is why the vast majority of data benchmarks' metrics are traditionally response time and throughput, as in *de facto* standard TPC benchmarks[8] or variants. Moreover, dealing with time or frequency as a metric is quite relevant, since it embeds I/Os, RAM and CPU altogether.

We also omitted metrics that are casual in textual applications, i.e., precision, recall and F1 measure, because they are not applicable to all the tasks we defined in Section 3.2. Moreover, we cannot compute such metrics, because they require data to be labeled, which is not the case of our benchmark's dataset. Finally,

[8] `https://www.tpc.org/`

as our dataset is dynamically generated with respect to the scale factor SF, it is impossible to predict what document is included at each execution. Therefore, we cannot provide in advance a ground truth to support precision, recall or F1.

3.4 Assessment Protocol

The three metrics from Section 4.2 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[5].
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. However, for comparison needs, we prescribe the following integration tasks:
   (a) transform tabular files into DBMS tables;
   (b) transform textual documents into vector format, e.g., a document embedding and/or a vector of term frequencies);
   (c) index all documents.
3. **Metadata size and generation time computing** consists in measuring 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). Caches must also be cleaned after each query run.

Algorithm 1: DLBench+ assessment protocol

```
Input: SF
Output: 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);
            clean_caches();
        end
        metric_1[SF][i] ← average(response_times);
    end
end
```

4 Qualitative Benchmarking

To complement the quantitative benchmark we propose (Section 3), we introduce in DLBench+ a qualitative part that consists to measure the user experience. To this aim, we reuse and complement existing approaches from the literature. We propose to measure the user experience on a data lake system through three metrics, namely usability, time-saving and usefulness.

4.1 Assessment Protocol

To measure the three qualitative metrics we introduce (usability, time-saving and usefulness), we propose a three-step protocol. The number of users should be as high as possible. It is also important to vary user profiles to include both business users (having little data management skills) and data management specialists. For each user, the three metrics are measured following this protocol [9].

1. **Getting started with the system**: The user is given a tutorial as well as a set of exercises to perform on the system, in a guided manner. Such tasks must reflect the main features of the system. For the systems featuring textual and tabular document management, the tasks we identify in Section 3.2 must serve as references (term-based search, finding related documents, top-keywords extraction, etc.). The allotted time must be extended until the user becomes autonomous.
2. **Work in autonomy**: In this step, the user is assigned with a set of tasks to perform autonomously. As previously, such tasks must also reflect the main features of the system. Moreover, a supervisor must verify that users correctly execute the assigned tasks.
3. **Response to a questionnaire**: In this last step, the user answers an anonymous questionnaire to evaluate his or her experience on the system, indirectly providing his or her assessment of the three metrics.

Once this process has been applied to each user, the three metrics can be deduced from the questionnaire's responses. A detailed example is provided in Section 4.2.

4.2 Performance Metrics

In this section, we define the three metrics we propose to qualitatively evaluate data lake systems.

Usability The notion of usability is quite abstract. It is thus difficult to evaluate and even more difficult to measure. Nevertheless, there are standard approaches that are commonly used and accepted in the academic and industrial literature to answer this need. We adopt the SUS approach that is widely known and used. It was indeed been used as a post-evaluation methodology in about 43% of IT projects in 2009 [21]. Moreover, it has already been used to evaluate the usability of a data lake system [1].

The SUS evaluation methodology consists in proposing a questionnaire to the users and then aggregating the answers into an usability score [5]. The higher the score, the more usable the system is considered to be, and vice versa. A typical questionnaire consists of 10 statements that alternately target the positive and negative points of the system being evaluated (Table 3). Users respond to each statement by assigning a value (rating) between 1 and 5, depending on whether they agree (5 for total agreement) or disagree (1 for total disagreement). The 10 answers are translated into a usability score ranging between 0 and 100, using Formulas 1, 2 and 3.

$$positive\_score = \sum_{i=1}^{9,\ step=2} (value_i - 1) \tag{1}$$

$$negative\_score = \sum_{i=2}^{10,\ step=2} (5 - value_i) \tag{2}$$

$$usability\_score = (positive\_score + negative\_score) \times 2.5 \tag{3}$$

A better interpretation of the usability score is provided by Bangor et al., who introduce a correspondence between the score and qualifiers by conducting a survey over 1000 applications and establishing a correlation between qualifiers and the corresponding average usability scores [3]. Thence, data lake systems can be classified into six groups of usability qualifiers (Table 4).

[9] `https://github.com/Pegdwende44/DLBench/blob/main/UserManual.md`

Table 3: SUS usability measure statements [5]

Order	Question
#1	I think that I would like to use this system frequently.
#2	I found the system unnecessarily complex.
#3	I thought the system was easy to use.
#4	I think that I would need the support of a technical person to be able to use this system.
#5	I found the various functions in this system well integrated.
#6	I thought there was too many inconsistencies in this system.
#7	I would imagine that most people could learn to use this system very quickly.
#8	I found the system very awkward (hard/difficult) to use.
#9	I felt very confident using the system.
#10	I needed to learn a lot of things before I could get going with this system.

Table 4: Usability score interpretation table [3]

Qualification	Mean score
Worse imaginable	12.5
Awful	20.3
Poor	35.7
OK	50.9
Good	71.4
Excellent	85.5
Best imaginable	90.9

Time-saving and usefulness The criteria of time-saving and usefulness we propose are inspired by the work of Fernandez et al., who use these notions to evaluate a data lake implementation [9]. Time-saving refers to the difference between the time required to perform a task using the system and the time required for the same task in a traditional approach, i.e., fully manually or using the tools and techniques previously known by the user. Similarly, the notion of usefulness reflects the quantity of knowledge that is now accessible thanks to the system, compared to a so-called traditional approach.

To measure time-saving and usefulness on a data lake, we propose to collect the users' appreciation through a questionnaire. For each criterion (time-saving and usefulness), questions are formulated with respect to the main features of the system. While evaluating the system, the user is thus expected to associate an appreciation ranging from 1 (extremely negative) to 5 (extremely positive) with each feature of the system. Results from the questionnaire are then aggregated to determine whether the system is considered effective in terms of time-saving and usefulness.

We introduce a typical questionnaire dedicated to data lake systems with textual and tabular document management capabilities in Table 5. The questions we propose are in line with the main features we identified in data lakes (Section 3.2).

5 Proof of Concept

5.1 Overview of AUDAL

To demonstrate the use of DLBench+, we evaluate AUDAL [32], a data lake system designed as part of a management science project that allows 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...). AUDAL uses an extensive metadata system stored inside MongoDB[10], Neo4J[11], SQLite[8] and ElasticSearch[12] to support numerous analyses. MongoDB stores refined textual docu-

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

Table 5: DLBench+ time-saving and usefulness questionnaire

Order	Question
	Time-saving
#1	Do category-based searches save you time compared to performing them on your own?
#2	Do keyword-based searches save you time compared to performing them on your own?
#3	Does the highlights feature save you time compared performing it on your own?
#4	Does the document scoring feature save you time compared to performing it on your own?
#5	Does top keyword retrieval save you time compared to performing it on your own?
#6	Does the document clustering feature save you time compared to performing it on your own?
#7	Does the SQL querying feature save you time compared to performing it on your own?
#8	Does the tuple clustering feature save you time compared to performing it on your own?
	Usefulness
#9	Do category-based searches provide you with useful information about the data?
#10	Do keyword-based searches provide you with useful information about the data?
#11	Does the highlights feature provide you with useful information about the data?
#12	Does the document scoring feature provide you with useful information about the data?
#13	Does top keyword retrieval provide you with useful information about the data?
#14	Does the document clustering feature provide you with useful information about the data?
#15	Does the SQL querying feature provide you with useful information about the data?
#16	Does the tuple clustering feature provide you with useful information about the data?

ments in the form of bag-of-words while SQLite analogously manages tabular document in a ready-to-query format. ElasticSearch manages indexes which support keyword-based searches and Neo4J stores relationships between documents. AUDAL provides ready-to-use analyses via a REpresentational State Transfer Application Programming Interface (REST API) dedicated to data scientists, but also through a Web-based analysis platform designed for business users (Figure 2).

AUDAL is implemented on a cluster of 3 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 3 VMs hosts a Neo4J instance, an ElasticSearch instance and a MongoDB instance to store AUDAL's metadata.

5.2 Quantitative Evaluation: System Performance

In this section, we present and discuss the quantitative benchmarking results achieved on AUDAL with DLBench+.

Metric #1: Query execution time We first analyze the response times obtained on the ten tasks defined in DLBench+. Figure 3 shows how the execution times of category filters evolve with SF on AUDAL. We notice a linear evolution on queries with one or two filtering axes (queries Q1A and Q1B). This evolution seems quadratic when the query is more complex (query Q1C). Response times are still acceptable with a response of 6.2 seconds for the most complex query when $SF = 5$, i.e., the maximum SF. In the context of the AURA-PMI project, users tolerate a few seconds of delay, but not a minute.

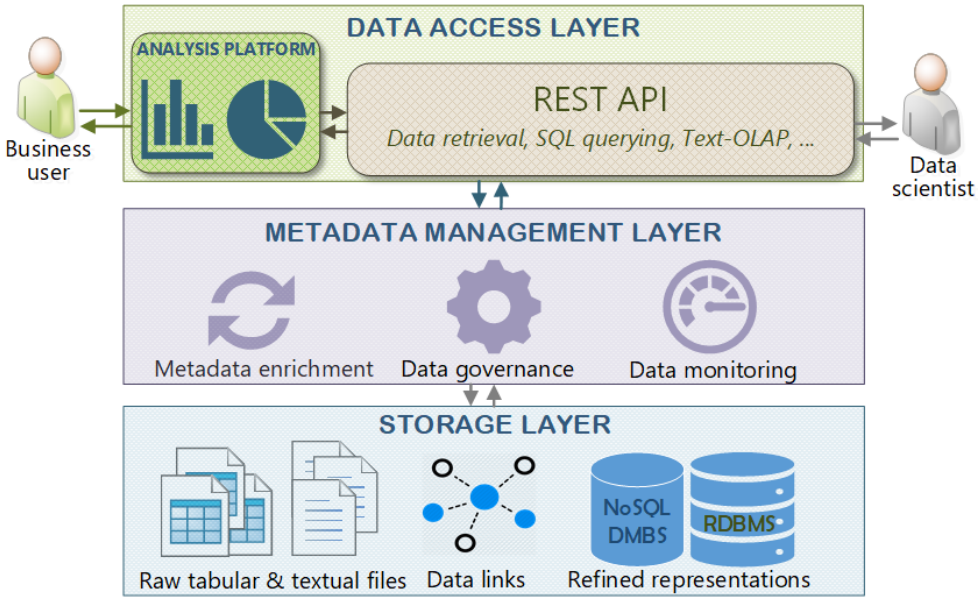

Fig. 2: Architecture of AUDAL [32]

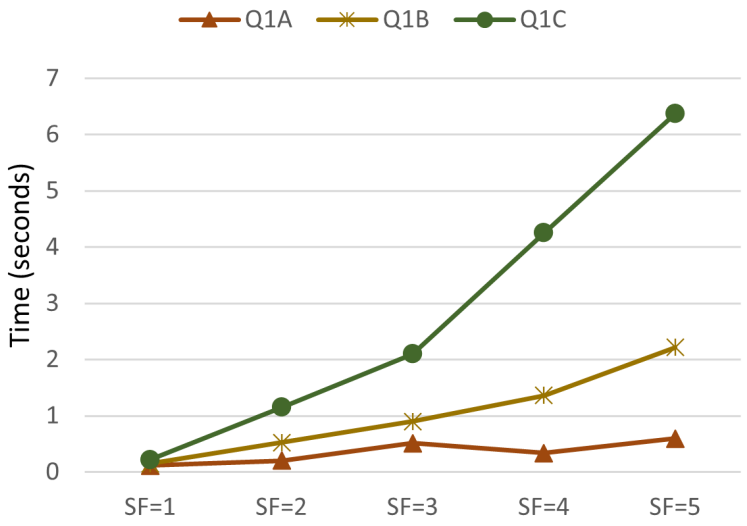

Fig. 3: Task #1 query execution time

Figure 4 shows the evolution of keyword-based search queries' response times. We observe a general linear trend with little evolution for both uni-term search (query Q2A), multi-term search (query Q2B) and fuzzy search (query Q2C). The evolution of response times seems noisy in some places (as can be seen from the irregular curve). This could be explained by the influence of external factors that are difficult to control, such as the traffic on the network hosting the virtual machines and the automatic memory optimization (garbage collection) performed by Java. This influence is however very weak (of the order of a tenth of a second) and is therefore only seen on results with very short response times, as is the case here.

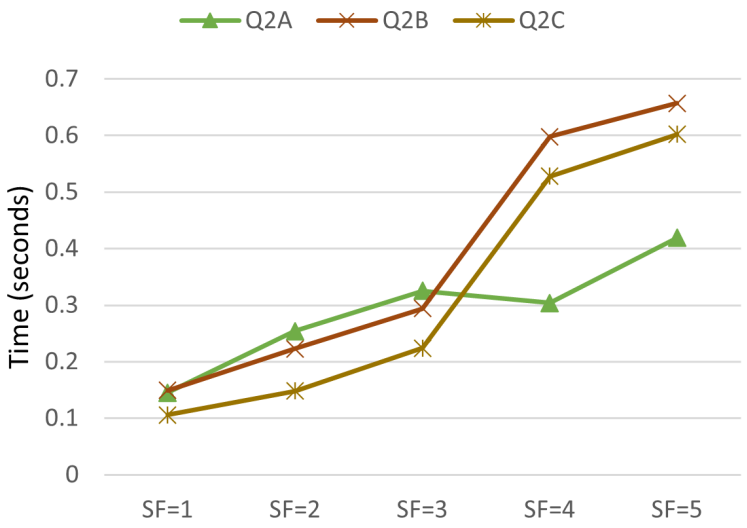

Fig. 4: Task #2 query execution time

Figure 5 shows the evolution of data search queries' response times. In addition to being very short, response times seems to evolve weakly for both text documents (query Q3A) and tabular documents (query Q3B).

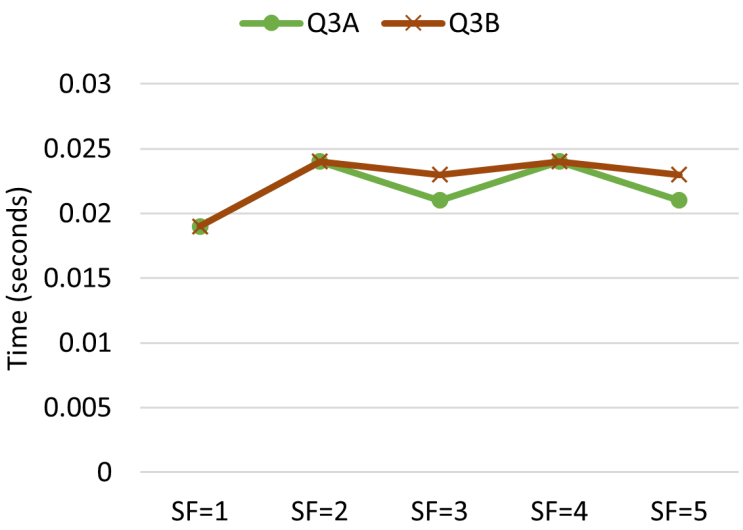

Fig. 5: Task #3 query execution time

Figure 6 shows the evolution of document scoring queries' response times. This evolution seems linear or even logarithmic. Again, response times are very short (all less than 0.19 seconds), which causes the appearance of unexpected variations.

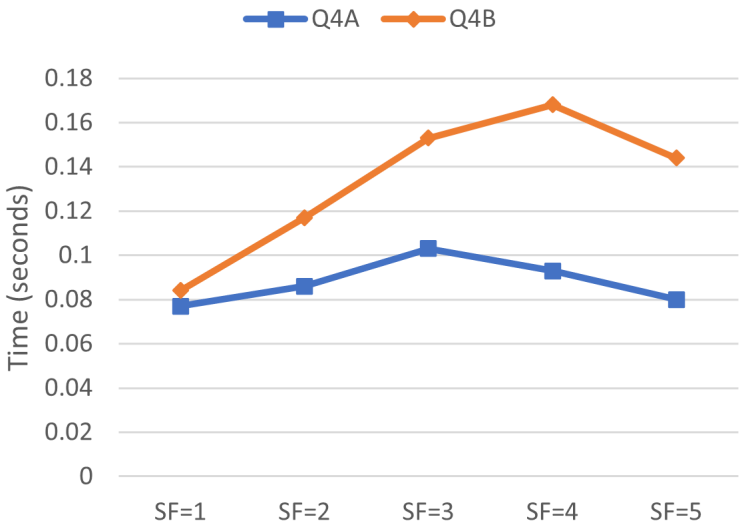

Fig. 6: Task #4 query execution time

Figure 7 shows the evolution of document highlights extraction response times. The observed response times again remain very short (less than 0.12 seconds) and more or less constant with SF. We also notice that there is no clear difference in response times with various numbers of search terms (queries Q5A and Q5B, respectively), nor with fuzzy search (query Q5C).

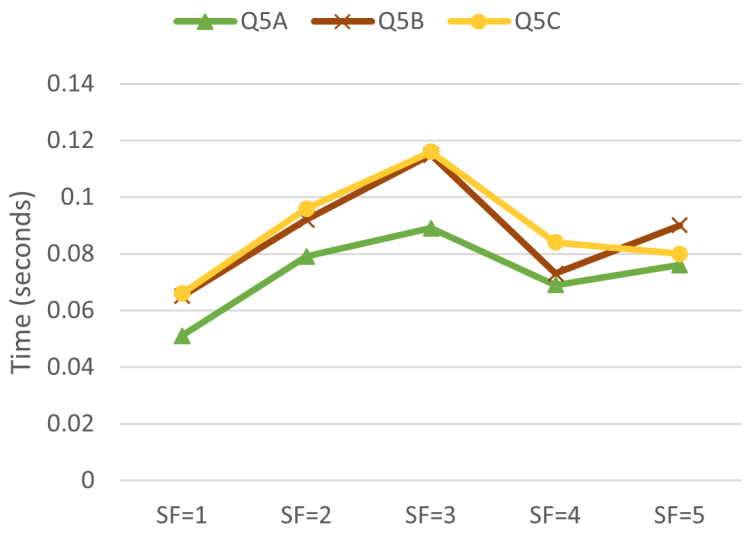

Fig. 7: Task #5 query execution time

Figure 8 shows the evolution of document top keywords counting response times. We observe a linear growth of the response times, which range from 18 seconds for $SF = 1$ to 79 seconds for $SF = 5$. Such response times are much higher than in the previous tasks. Even the linear evolution makes this result acceptable, the response times need to be improved to avert exceeding the one minute threshold as it is with the higher scale factors.

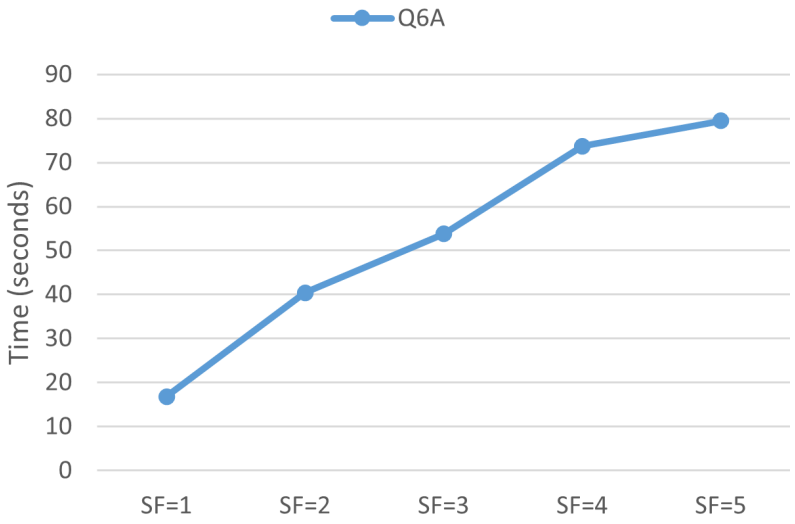

Fig. 8: Task #6 query execution time

Figure 9 shows the evolution of document text-mining response times, which evolve linearly with SF. Moreover, both types of clustering (PCA and KMeans) have almost identical execution times, ranging from 4 seconds for $SF = 1$ to 21 seconds for $SF = 5$. Such response times remain in the order of a second and are therefore acceptable to users.

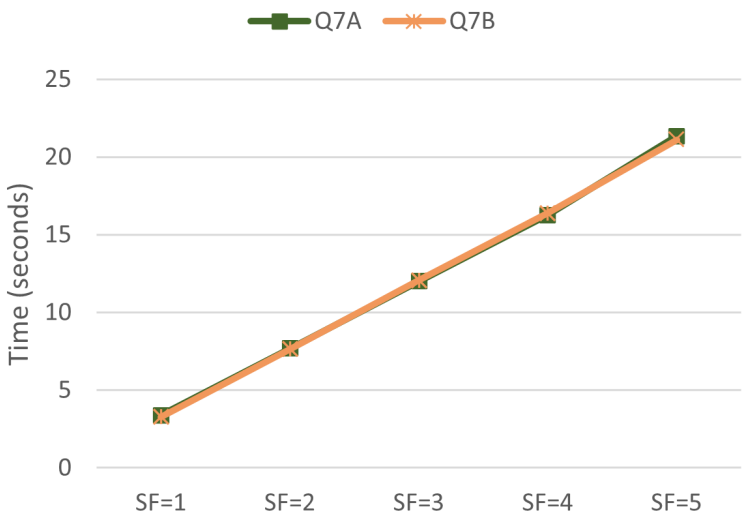

Fig. 9: Task #7 query execution time

Figure 10 shows the evolution of simple table queries' response times. In our case, these are SQL queries. We observe that response times do not seem to evolve with SF. The observed evolution seems to come mainly from the noise generated by the external factors, as previously mentioned.

We particularly note that query Q8A takes on average 0.18 seconds to run, while query Q8C only runs in 0.02 seconds. As the two queries only vary with respect to the quantity of data returned (all tuples for Q8A and the count for Q8C)), we can conclude that the 0.16 seconds difference on average is due to I/Os induced by additional result volume.

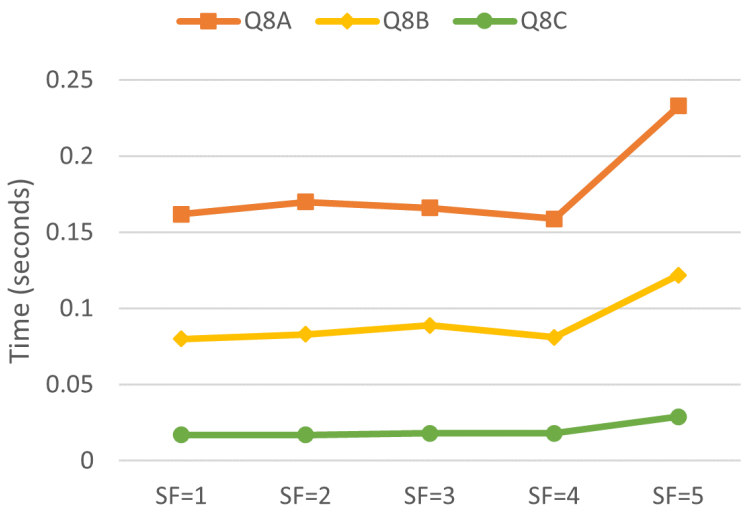

Fig. 10: Task #8 query execution time

Figure 11 shows the evolution of complex table queries' response times. These are join and SQL aggregation queries. The results show very short response times (less than 0.08 seconds), surprisingly lower than

in the case of simple queries, certainly due to a lower number of processed tuples. The only variations seem to be related to external influences.

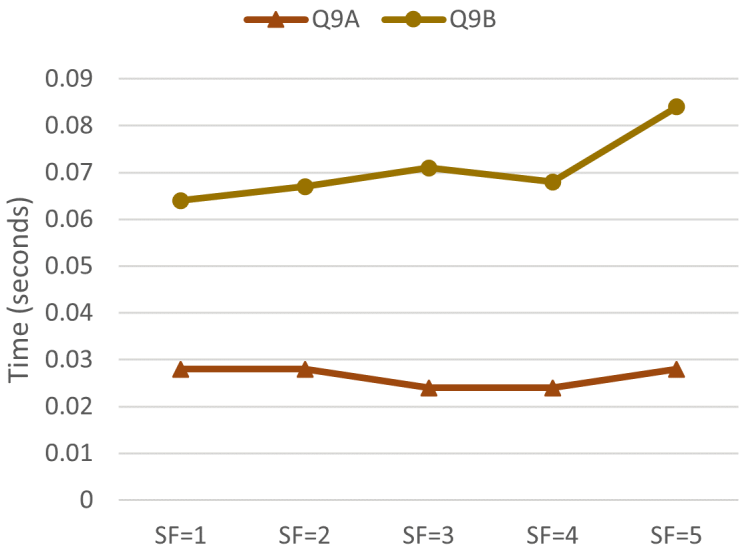

Fig. 11: Task #9 query execution time

Eventually, Figure 12 shows the evolution tuple mining queries' response times. Again, response times are short and remain constant with SF. However, we notice that the PCA algorithm (query Q10A, 0.16 seconds on average) runs much faster than the KMeans algorithm (query Q10B, 0.51 seconds on average). This is probably due to the difference in complexity between the two algorithms (PCA and KMeans). The complexity of KMeans indeed depends on the number n of tuples, while the complexity of PCA depends on the number p of columns. In our case, p is presumably much lower than n, which could explain the better performance achieved by PCA.

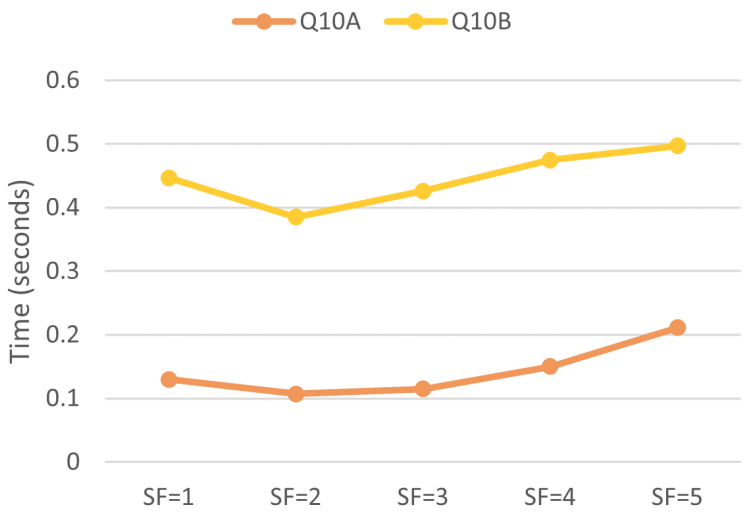

Fig. 12: Task #10 query execution time

Metric #2: Metadata size The size of AUDAL's metadata seems to evolve logarithmically with SF, unlike the size of the raw data, which evolves linearly. While comparing the size of metadata and raw data, we indeed observe that the ratio of metadata to data decreases with SF (Figure 13). As a proof, the size of the metadata goes from 83% for $SF = 1$ to 54% of the raw data for $SF = 5$.

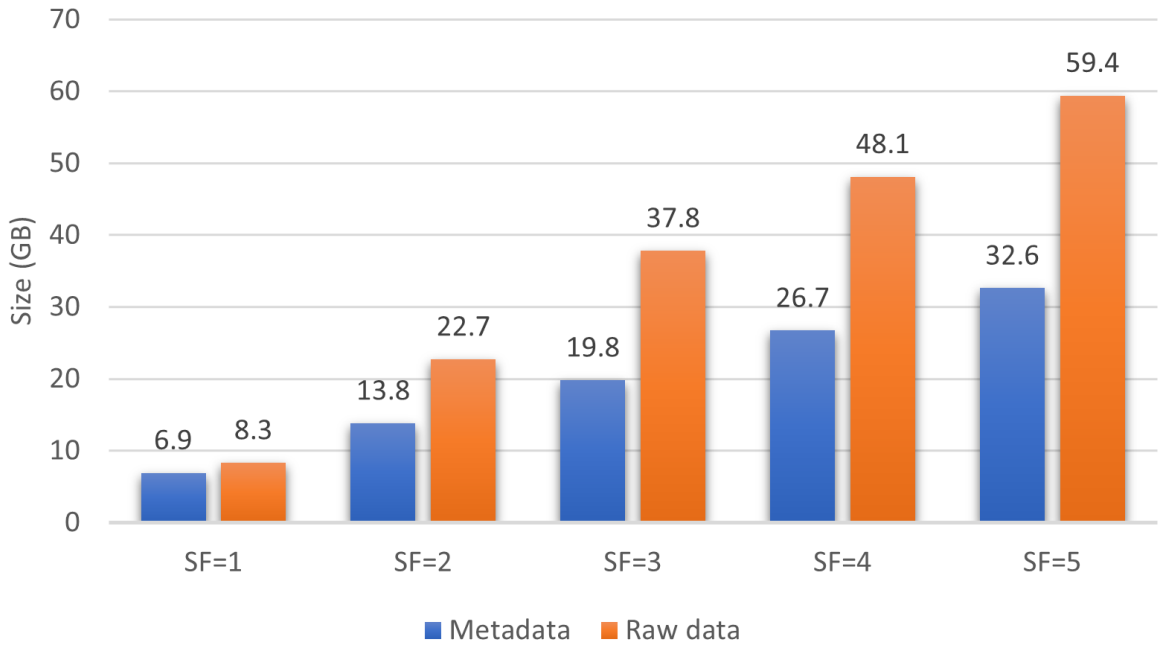

Fig. 13: Evolution of metadata size

Metric #3: Metadata generation time As shown in Figure 14, the time necessary to generate AUDAL's metadata system from scratch seems to evolve exponentially. It ranges from 3.4 hours for $SF = 1$ to 72 hours for $SF = 5$. More than the generation time itself, it is its evolution that is unsatisfactory, because it shows a difficulty to scale up. Possible solutions to this limitation could be in a better and further distribution of processes on the machines. Another way to address this could be to translate all of Python's metadata generation processes into a lower-level programming language such as C or C++. This would allow for more efficient resource management.

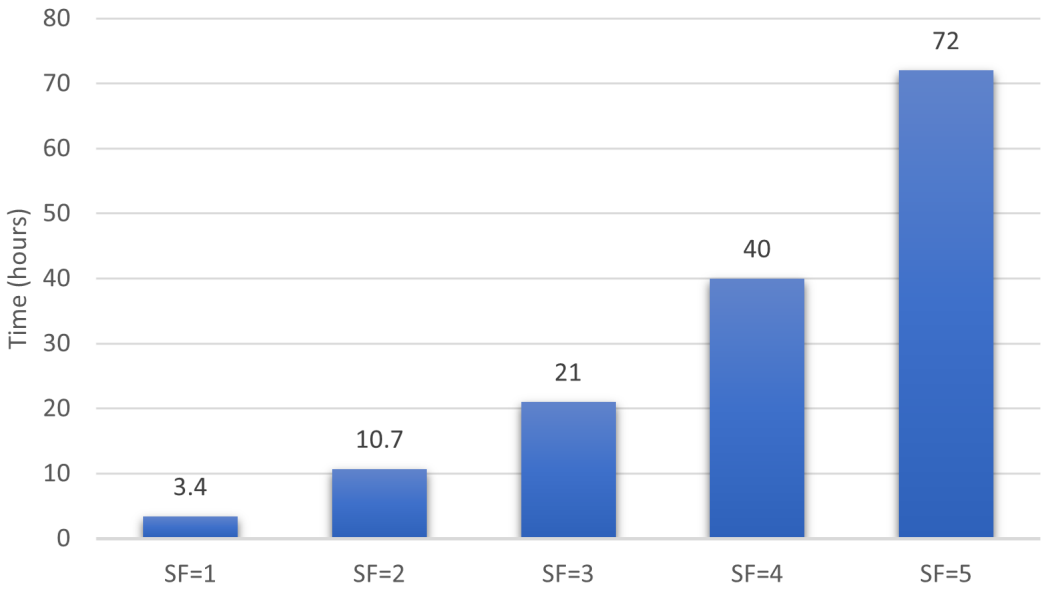

Fig. 14: Evolution of metadata generation time

5.3 Qualitative Evaluation: User Experience

In addition to the quantitative evaluation, we also run DLBench+'s qualitative part to evaluate the user experience on AUDAL. We follow the protocol from Section 4.1 to get a feedback from 6 users. We called for such a small number of users because the evaluation protocol is time consuming, i.e., about two hours for both one user and the system administrator. Yet, we notice that the Aurum data lake [9] was only been evaluated by 4 users, presumably encountering the same issues as us.

Moreover, the users' feeling is that interacting with the application for 75 minutes is no enough for fully handle the system. In such conditions, users might perceive no the system as usable as it is. Unfortunately, we could not do otherwise, since the evaluation protocol is already time consuming and the agenda of potential users quite full.

Eventually, all six users participating in the evaluation process are all PhD students and professors working for the same project. Such proximity might incur biases. To tackle this issue and make their appreciation as objective as possible, we anonymized the questionnaire.

The number of participants might seem very low. However, qualitative evaluation is time-consuming for users. It indeed requires each user to spend several hours on the system before answering the questionnaire. Moreover, we are not the only ones to encounter this problem. Fernandez et al. made their data lake system assessed by only three users [9], while the DATAMARIAN system has been evaluated by six users [13].

On the basis of the users' responses, we evaluate the user experience on AUDAL by assessing the three metrics (usability, time-saving and usefulness) defined in Section 4.2.

Metric #1: Usability By applying the usability calculation formula, we obtain an average usability score of 65 for AUDAL. Referring to Bangor et al.'s qualifiers [3] (Table 4), we notice that AUDAL can be considered to have a "good" usability , as its score close to the average score corresponding to this qualifier. However, we note that AUDAL's usability score is lower than that of Bagozi et al.'s data lake system [1] (87.5).

Yet, based on the work of Kortum [20], we can compare AUDAL's usability score to widely-used applications. Figure 15 shows that AUDAL is less usable than Microsoft Word[13] (76.2 usability score), but more usable than Microsoft Excel[14] (56,5).

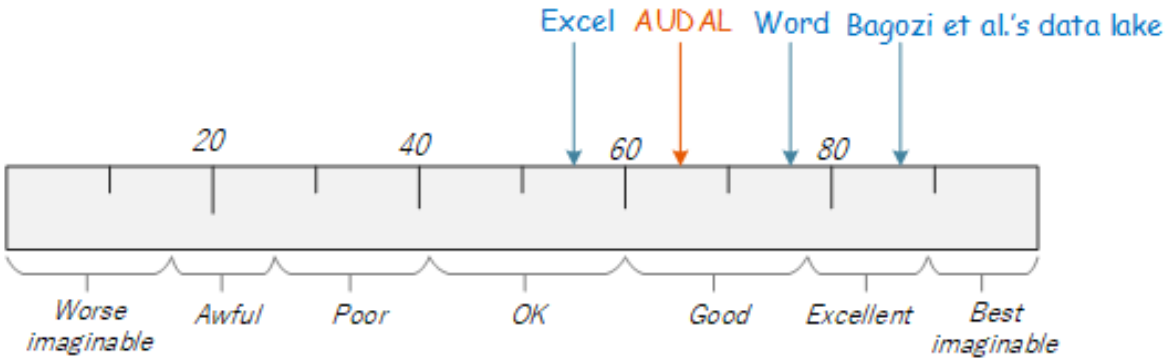

Fig. 15: Comparison of usability scores

Metrics #2: Time-saving Figure 16 presents the distribution of user responses about time-saving. Scores lie on the Y-axis. As a reminder, the minimum score (1) expresses a total absence of time-saving, while the maximum score (5) expresses an absolute time-saving. From this distribution, we can interpret that AUDAL induces a significant time-saving according to the users. The average score (4.23) is indeed higher than the average expected from a "neutral" system (3).

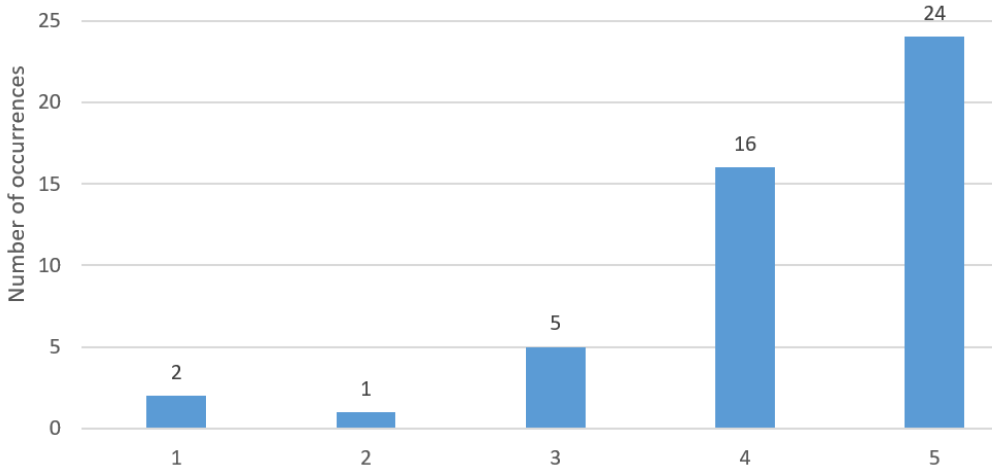

Fig. 16: Time-saving response distribution

[13] urlhttps://www.microsoft.com/fr-fr/microsoft-365/word
[14] urlhttps://www.microsoft.com/fr-fr/microsoft-365/excel

To statistically validate this observation, we compare the average of the scores obtained to the value corresponding to a neutral system. We verify that this overall positive assessment is significant through a mean comparison Student's t-test [22]. This statistical test allows us to compare an average observed on a set of values (scores, in our case) with a theoretical value (the neutral value 3, in our case). The null hypothesis H0 and the alternative hypothesis H1 of this test are formulated as follows.

- H0: the average score is equal to 3.
- H1: the average score is significantly higher than 3.

The p-value obtained at the end of the test is 0.0001 and is thus lower than the standard error risk of 5%. We therefore reject the null hypothesis and conclude that the average time-saving score on of AUDAL significantly higher than 3.

Metrics #3: Usefulness Figure 17 presents the distribution of user responses about usefulness. Again, scores lie on the Y-axis. The general trend shows that, in the vast majority of cases, users find AUDAL globally useful. We indeed note an average score of 3.85, which is higher than the average expected from a neutral system. This observation is also statistically confirmed by a Student's t-test producing a p-value of 0.0001, which is lower than the standard error risk of 5%. We thus conclude that the average usefulness score of AUDAL is significantly higher than 3, too.

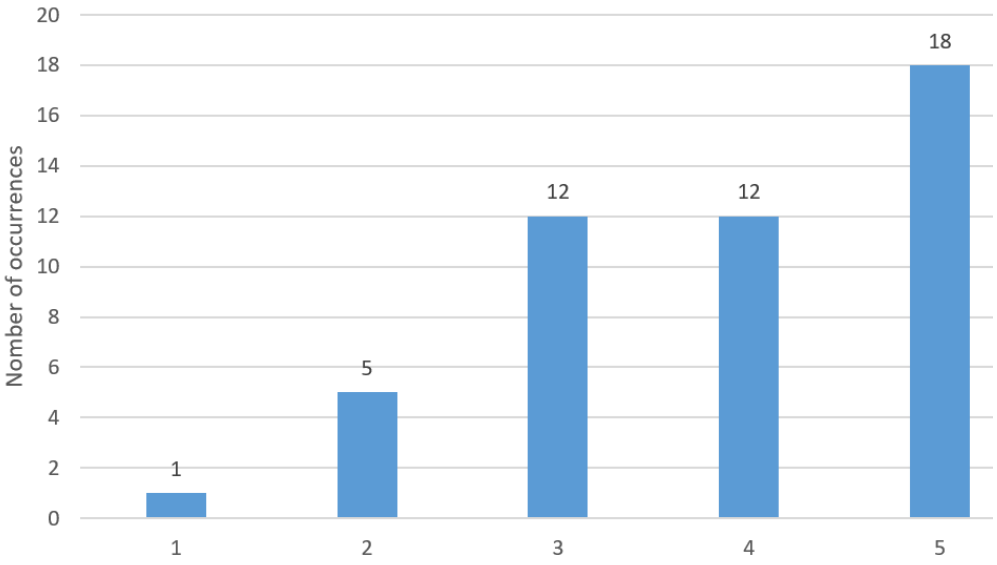

Fig. 17: Usefulness response distribution

6 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 featuring quantitative and qualitative evaluation.

DLBench+ features in its quantitative side:

1. a data model made of a corpus 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;
4. an execution protocol.

In its qualitative side, DLBench+ includes another set of three metrics and a survey-based evaluation protocol to measure the user experience.

As a proof of concept, we show the use of DLBench+ by assessing the AUDAL data lake system [32]. The quantitative evaluation of system performance highlights that the AUDAL system scales quite well, especially for data retrieval and tabular data querying.

All in all, the AUDAL data lake system can be considered as globally effective. It indeed shows some satisfactory results regarding both the quantitative and qualitative assessments.

From the quantitative assessment, we can note that all proposed tasks run within reasonable times. The longest task (Task #6) takes less than 1.5 minute to run. This is acceptable to AUDAL's users from the qualitative assessment's results. We also note that query response times remain constant or evolve linearly with the scale factor. Queries from task #6 thus take between 0.19 and 0.24 seconds to execute, whatever the scale factor. We consider them as constant as the execution time seems not linked to the scale factor. A *contrario*, query response times form task #1 clearly depend on the scale factor, with for example the query Q2B taking 0.15 seconds for the lowest scale factor (SF=1) and 2.21 seconds for the highest scale factor (SF=5).

However, this comes at the cost of a broad data integration task and an expensive metadata system. Metadata size measurements indeed show that metadata accounts for at least half of the raw data size when $SF = 5$, and even 83% of data volume when $SF = 1$. Yet, the real weakness underlined by AUDAL's quantitative assessment is the time taken for metadata generation, i.e., this metric evolves exponentially and is therefore not sustainable.

Fortunately, our qualitative evaluation still hints that AUDAL provides a better user experience than one would expect from an average system.

Future works include performing a comparative study of existing data lake systems using DLBench+. However, since AUDAL is the only data lake system that supports both textual and tabular data, to the best of our knowledge, we cannot compare it to any equivalent system.

Thence, at this time, we resort to propose alternative implementations of AUDAL, e.g., re-implementing AUDAL with Cassandra instead of MongoDB, Apache Solr instead of Elasticsearch, Allegrograph instead of Neo4J and MariaDB instead of SQLite, and all combinations of these software components.

Another lead is, since unstructured data are currently seldom included in data lakes, to focus on the structured part and user-experience evaluation to compare data lake systems such as Aurum [9], Mami et al.'s [24] or Bogatu et al.'s systems [4].

Another enhancement of DLBench+ could consists in providing an overview of value distributions in generated data. That would allow a further analysis of the evaluation results, especially for keyword-related tasks. For example, we could analyze how the system reacts to a query with very frequent or uncommon terms. The results of these in-depth analyses could be used to improve the indexing strategy underlying the system.

Finally, we also envisage an extension of the structured part of DLBench+'s data model with an alternative, larger dataset. In this way, we would better reflect the challenge related to data volume in the structured part of our benchmark.

Acknowledgments

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

References

1. Bagozi, A., Bianchini, D., Antonellis, V.D., Garda, M., Melchiori, M.: Personalised Exploration Graphs on Semantic Data Lakes. In: On the Move to Meaningful Internet Systems (OTM 2019), Rhodes, Greece. pp. 22–39 (October 2019). https://doi.org/10.1007/978 − 3 − 030 − 33246 − 4_2
2. 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

3. Bangor, A., Kortum, P., Miller, J.: Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of usability studies **4**(3), 114–123 (2009)
4. Bogatu, A., Fernandes, A., Paton, N., Konstantinou, N.: Dataset Discovery in Data Lakes. In: 36th IEEE International Conference on Data Engineering (ICDE2020), Dallas, Texas, USA (April 2020)
5. Brooke, J.: Sus: a quick and dirty usability scale. Usability evaluation in industry **189** (1996)
6. Darmont, J.: Data-centric benchmarking. In: Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics, pp. 342–353. IGI Global (2019)
7. Dixon, J.: Pentaho, Hadoop, and Data Lakes (October 2010), `https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/`
8. Elasticsearch: Theory Behind Relevance Scoring (September 2019), `https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html`
9. Fernandez, R.C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., Stonebraker, M.: Aurum: A Data Discovery System. In: 34th IEEE International Conference on Data Engineering (ICDE 2018), Paris, France. pp. 1001–1012 (April 2018)
10. Ferrarons, J., Adhana, M., Colmenares, C., Pietrowska, S., Bentayeb, F., Darmont, J.: Primeball: A parallel processing framework benchmark for big data applications in the cloud. In: Nambiar, R., Poess, M. (eds.) Performance Characterization and Benchmarking. pp. 109–124. Springer International Publishing, Cham (2014)
11. 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
12. Gao, W., Zhan, J., Wang, L., Luo, C., Zheng, D., Ren, R., Zheng, C., Lu, G., Li, J., Cao, Z., Zhang, S., Tang, H.: Bigdatabench: A dwarf-based big data and AI benchmark suite. CoRR **abs/1802.08254** (2018), `http://arxiv.org/abs/1802.08254`
13. Gao, Y., Huang, S., Parameswaran, A.: Navigating the data lake with datamaran: Automatically extracting structure from log datasets. In: 2018 International Conference on Management of Data. pp. 943–958 (2018)
14. 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
15. 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`
16. Gray, J.: Database and transaction processing performance handbook. (1993), `http://jimgray.azurewebsites.net/benchmarkhandbook/chapter1.pdf`
17. 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
18. 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
19. 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
20. Kortum, P.T., Bangor, A.: Usability ratings for everyday products measured with the system usability scale. International Journal of Human-Computer Interaction **29**(2), 67–76 (2013)
21. Lewis, J.R.: The system usability scale: past, present, and future. International Journal of Human–Computer Interaction **34**(7), 577–590 (2018)
22. Livingston, E.H.: Who was student and why do we care so much about his t-test? 1. Journal of Surgical Research **118**(1), 58–65 (2004)
23. 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
24. Mami, M.N., Graux, D., Scerri, S., Jabeen, H., Auer, S.: Querying Data Lakes using Spark and Presto. In: Proceedings of the 2019 World Wide Web Conference (WWW'19), San Francisco, CA, USA (May 2019)
25. 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

26. Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.R., Galinari, P.: Lshtc: A benchmark for large-scale text classification. arXiv preprint arXiv:1503.08581 (2015)
27. 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
28. Ravat, F., Zhao, Y.: Data Lakes: Trends and Perspectives. In: 30th International Conference on Database and Expert Systems Applications (DEXA 2019), Linz, Austria (August 2019)
29. Reagan, A.J., Tivnan, B., Williams, J.R., Danforth, C.M., Dodds, P.S.: Benchmarking sentiment analysis methods for large-scale texts: a case for using continuum-scored words and word shift graphs. arXiv preprint arXiv:1512.00531 (2015)
30. Russom, P.: Data Lakes Purposes, Practices, Patterns, and Platforms. TDWI research (2017)
31. Sawadogo, P.N., Darmont, J., Noûs, C.: Benchmarking Data Lakes Featuring Structured and Unstructured Data with DLBench. In: Proceedings of the 23rd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2021), Linz, Austria ((to appear) 2021)
32. Sawadogo, P.N., Darmont, J., Noûs, C.: Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. In: Proceedings of the 25th European Conference on Advances in Databases and Information Systems (ADBIS 2021), Tartu, Estonia ((to appear) 2021)
33. Simitsis, A., Vassiliadis, P., Dayal, U., Karagiannis, A., Tziovara, V.: Benchmarking ETL workflows. In: Nambiar, R.O., Poess, M. (eds.) Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5895, pp. 199–220. Springer (2009). https://doi.org/10.1007/978-3-642-10424-4_15, `https://doi.org/10.1007/978-3-642-10424-4_15`
34. Steinley, D.: K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology **59**(1), 1–34 (2006)
35. TPC: TPC Benchmark H - Standard Specification (version 2.18.0). http://www.tpc.org/tpch/ (2014)
36. TPC: TPC Express Benchmark HS - Standard Specification (version 2.0.3). http://www.tpc.org/tpcds/ (2018)
37. TPC: TPC Benchmark DS - Standard Specification (version 2.13.0). http://www.tpc.org/tpcds/ (2020)
38. TPC: TPC Express AI - Draft Specification (version 0.6). http://tpc.org/tpcx-ai/default5.asp (2020)
39. 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
40. 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)
41. 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
42. Wyatt, L., Caufield, B., Pol, D.: Principles for an ETL benchmark. In: Nambiar, R.O., Poess, M. (eds.) Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5895, pp. 183–198. Springer (2009). https://doi.org/10.1007/978-3-642-10424-4_14, `https://doi.org/10.1007/978-3-642-10424-4_14`
43. 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