PRIMEBALL:
a Parallel Processing Framework Benchmark for Big Data Applications in the Cloud

Jaume Ferrarons, Mulu Adhana, Carlos Colmenares, Sandra Pietrowska,
Fadila Bentayeb and Jérôme Darmont

Université de Lyon (Laboratoire ERIC)
Université Lumière Lyon 2 – 5 avenue Pierre Mendès-France
69676 Bron Cedex – France
first_name.last_name@univ-lyon2.fr

Abstract. In this paper, we draw the specifications of a novel benchmark for comparing parallel processing frameworks in the context of big data applications hosted in the cloud. We aim at filling several gaps in already existing cloud data processing benchmarks, which lack a real-life context for their processes, thus losing relevance when trying to assess performance for real applications. Hence, we propose a fictitious news site hosted in the cloud that is to be managed by the framework under analysis, together with several objective use case scenarios and measures for evaluating system performance. The main strengths of our benchmark are parallelization capabilities supporting cloud features and big data properties.

1   Introduction

We are currently living through an information revolution that has undoubtedly brought a massive increase in the volume of data being produced and stored worldwide. In this Internet age, where the world creates 2.5 exabytes of data every day [1], traditional approaches and techniques for data analysis proved limited because some lack parallelism, and most lack fault tolerance capabilities. Therefore, in recent years, many platforms for parallel processing have been created so as to satisfy this need. These platforms provide frameworks for storing, accessing, updating and deleting data efficiently in computer clusters, ensuring fault tolerance and making the whole process transparent to users. Examples of such systems include Google's BigQuery [2] and Apache's Hadoop [3].

In this context, the terms "big data" are used for referring to digital information that comes in high volume, velocity and variety [1]; and the systems that make use of this type of data for achieving profitable objectives can be referred to as big data applications. Several examples of big data applications can be found

in the areas of capital market, risk management, retail, social media analysis and meteorology. This kind of applications, beside requiring high parallel processing capabilities for analysis, also needs a good and scalable infrastructure capable of adapting quickly to an increment in computing or storage needs. Therefore, many big data applications are being deployed in the cloud so as to allow fast adaptability and flexibility.

Given the recent increase of big data applications in the cloud, and the use of parallel processing frameworks for dealing with the technical issues implied by the use of clusters and large amount of complex data, it has become important to fix standards so as to allow accurate comparisons of these frameworks. Several benchmarks already exist for measuring a system's parallelization capabilities, cloud features or big data analysis abilities, but none of them offers direct means of accurately measuring: 1) the three of them 2) in a real-life context.

Thus, following the principles defined by Folkerts et al. [4], we propose the specifications of PRIMEBALL as a complete and unified benchmark for assessing a system's performance w.r.t. two main axes involved in the context of big data applications hosted in the cloud: parallel processing frameworks and cloud computing service providers. PRIMEBALL also aims to emulate common usages of cloud services, data manipulations and data transfers.

The remainder of this paper is organized as follows. Section 2 reviews existing benchmarks similar to PRIMEBALL and motivates its design. Section 3 provides an overview of PRIMEBALL. Then, Sections 4, 5 and 6 detail the specification of PRIMEBALL's components, i.e., its dataset, workload, and properties and metrics, respectively. Finally, Section 7 concludes this paper and provides future research leads.

2   Related Work

Among the standard TPC benchmarks, TPC-DS, a decision support benchmark that models several generally applicable aspects of a decision support system, including queries and data maintenance [5], is closely related to data analytics we target at. However, although it can generate high volumes of data, its underlying business model is a classical retail product supplier, thence its dataset could not fully qualify as big data-oriented because of a lack in structural variety.

MalStone is benchmark for data intensive computing and analysis [6]. It features MalGen, a synthetic data generator that produces large datasets to perform benchmarking. Data is designed to assess systems from the parallel processing point of view. Data is generated probabilistically following specified distributions.

Cloud Harmony measures the performance of cloud providers as black boxes [7]. The tests performed are mainly focused on assessing hardware performance or specific technologies. Cloud Harmony actually aggregates the results of benchmarks that existed before.

The Yahoo! Cloud Serving Benchmark (YCSB) is a framework to facilitate performance comparisons among cloud database systems [8] that mainly focuses

on key-value stores such as Dynamo [9]. YCSB defines several metrics and workloads to measure the behavior of the systems in different situations, or the same system when using different configurations.

Finally, the Statistical Workload Injector for MapReduce, or SWIM benchmark, is an open source benchmark that enables rigorous performance measurement of MapReduce systems [10]. It contains suites of workloads of thousands of jobs, with complex data, arrival, and computation patterns, and therefore provides workload-specific optimizations. SWIM is currently integrated with Hadoop.

We provide in Table 1 a synthetic comparison of all the above-mentioned benchmarks' properties, as well as PRIMEBALL's as a point of reference.

	TPC-DS	MalStone	Cloud Harmony	YCSB	SWIM	PRIMEBALL
Real data	∼	No	∼	No	No	Yes
Real workload	Yes	No	∼	No	Yes	Yes
Parallel processing	Yes	Yes	Yes	Yes	Yes	Yes
Hardware-oriented	No	No	Yes	No	No	No
MapReduce-oriented	No	No	No	No	Yes	∼
Cloud properties	∼	Yes	No	Yes	Yes	Yes
Complex data	No	No	Yes	No	Yes	Yes
Big data	∼	Yes	Yes	Yes	Yes	Yes
Technology-indep	Yes	Yes	Yes	No	No	Yes

Table 1. Comparison of Benchmark Features

The first property we compare is whether the data processed by a benchmark is produced artificially or extracted from a real environment. Only PRIMEBALL offers the possibility to work with wholly real data, with the aim to better simulate real applications. Although some of the benchmarks used in Cloud Harmony do use real data, most of them are actually processing artificial data. Proposing a real workload is the second property we selected. A benchmark system bearing this property is executing real-world operations to better simulate a production environment. PRIMEBALL, SWIM and TPC-DS execute only tasks that are closely related to real-world workloads. Moreover, some benchmarks in Cloud Harmony also execute tasks that are common in real environments.

The next property is satisfied by all analyzed benchmarks, i.e., they are all aimed at assessing parallel processing. By contrast, Cloud Harmony is the only benchmark that assesses the performance of specific pieces of hardware. For example, it has benchmarks for measuring CPU performance, memory I/O and disk I/O. The other benchmarks can give a notion of the performance of specific parts of the hardware, but are not that specific.

The MapReduce-related property refers to benchmarks aiming at measuring the quality of a system in terms of the performance obtained when executing

MapReduce tasks. SWIM is the only benchmark that is uniquely dedicated to MapReduce. However, if other benchmarks such as TPC-DS or PRIMEBALL are implemented using MapReduce tasks, they could measure performance through them too. Cloud properties refer to the prominent features of cloud computing. All benchmarks but TPC-DS and Cloud Harmony are designed to measure properties such as vertical and horizontal scalability, consistency, etc. (cf. Section 6.1). PRIMEBALL has been designed as well to be able to assess this kind of properties.

Complex data properties describe the benchmarks that are oriented to execute procedures using complex data structures to assess the system under test (SUT). TPC-DS implements a classical data warehouse with numerical and textual values. MalStone only aims to generate a big dataset as a log file and measures system response while processing it, thus the data processed is not complex. The same is true for YCSB, which assesses the performance of key-value stores. Values can be complex, but they are not processed, only stored. The other benchmarks, including PRIMEBALL, include complex data in some of their procedures. Big data properties describe the benchmark systems that involve analytical aspects over large amounts of data. All benchmarks have analytical situations involving large amounts of data. PRIMEBALL has specifically been designed to satisfy this property.

Finally, technology independence describes the systems the are designed to work with several kinds of technologies. YCSB and SWIM do not fulfill this property, because YCSB is oriented to analyze the performance of key-value stores only; and SWIM only assesses MapReduce procedures, and thus only makes sense when the SUT is able to execute them. The other benchmarks, including PRIMEBALL, can be used in environments that are not constrained by a given technology.

3 PRIMEBALL Overview

3.1 Application Model

PRIMEBALL's contextual application is set around New Pork Times: a fictitious on-line information service including international news, current affairs, documentaries, science, health and lifestyle sections. It is constantly updated and available 24-hours all around the world. New Pork Times hosts articles and multimedia documents about the latest news, as well as a large archive of past information.

All of these data are stored in a system called New Pork Times' News Hub (NPT-NH), which resides in a cluster hosted by some cloud service provider. This cluster is managed by a framework for parallel data processing and provides a remote storage that allows the user to access the files in the cluster without having to worry about their distribution in nodes. This storage system allows the user to insert and update data, and also to execute batch processes for analyzing/processing the data.

3.2 PRIMEBALL Features

This section lists what PRIMEBALL is/does and does not, so that its position, notably w.r.t. state of the art benchmarks (Section 2), is clear.

On one hand, PRIMEBALL:

− is a benchmark. It aims to compare the performance of the parallel processing framework under test with respect to several meaningful metrics;
− is cloud-oriented. The obtained results could also be used to compare:
  • cloud platforms as parallel processing frameworks,
  • service providers executing systems using the same cloud platform;
− is repeatable. All the proposed experiments are designed to lead to the same results if they are executed under the same conditions;
− does define a set of operations that is meaningful in the context of parallel processing and cloud computing;
− does define performance metrics that are oriented to measure cloud properties. The criteria to assess each metric are also defined;
− does define data relationships. We provide a description of the information stored in the SUT to be processed during the benchmark run.

On the other hand, PRIMEBALL does not:

− define technical execution details. It defines guidelines, but given that SUTs can be very different, the relevance of results is tightly related to implementation details;
− define expected performance results. No absolute value is provided as a comparison point, given that they are subject to implementation details;
− compare data retrieval or processing algorithms;
− define a storage schema. However, we define how data are physically stored.

Thence, PRIMEBALL is the first cloud-oriented unified benchmark aiming to assess all the elements involved in cloud-based big data application systems.

4 PRIMEBALL Dataset

For using PRIMEBALL, it is necessary to implement NPT-NH (Section 3.1). Therefore, the following subsections contain a technical description of its architecture, the type of data it contains and the operations performed onto data by means of batch processes, such as metadata extraction.

4.1 Types of Files to be Hosted

The system's database shall hold only three types of files. However, there can be many files of the same type. The three types of files follow.

– General information (XML): This set of files comprise the many XML documents that describe the standard information stored by NPT-NH, i.e., information about authors, the actual news, and so on. Section 4.2 describes the conceptual schema of this information.
– Media files (binary): Some articles make references to these files, which can be either audio or video documents.
– Metadata (XML): Several metadata for information retrieval and other tasks are extracted from the other two types of files by internal algorithms, for further querying. These metadata must be persisted as XML files in the system.

4.2 PRIMEBALL Schema

The system must hold as XML files data about the following entities:

– articles: the actual news articles;
– topics: the topics an article may belong to;
– keywords: sets of words that roughly describe the content of an article;
– languages: marks for indicating what language/dialect an article is written in;
– authors: people who write the articles;
– journalists: authors who work in journals and make interviews;
– professionals: specialists in some topic who write special analyses;
– countries: information about countries authors might be citizens of or work in;
– dates: information about the day of the year when an article was written;
– media: reference to a media file with some internal comments.

The conceptual schema of this dataset is featured in Figure 1. Its actual implementation depends on the framework for parallel data processing to be benchmarked and its capabilities for storing data.

4.3 Initial Data

To create an initial corpus for populating NPT-NH, PRIMEBALL shall come bundled with a crawler that extracts, transforms and loads information from any real world news hub akin to New Pork Times (a famous news site may come to mind). The crawler fetches information about news published during a requested period of time, which is recommended to be set up as the last 40 years. Moreover, it must also extract information about authors, media, and all relevant entities related to the system's architecture. Due the big data properties of the benchmark, it is recommended to fetch at least 100 TB of data for running the tests, and ideally 1 PB of information.

Once the corpus has been fetched, it can be sliced at will for selecting any scale factor for the initial data population. However, it is important to bear in mind that if the whole corpus is used in the initialization phase, then it will not be possible to perform updates or scale operations, since there will not be extra data available.
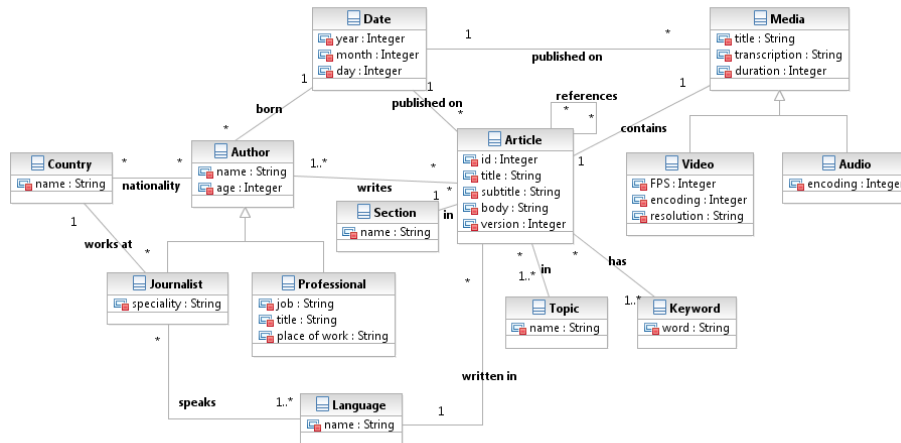
Fig. 1. Conceptual Schema of PRIMEBALL's Dataset

4.4 Metadata Processes

When data are loaded, it is necessary to run algorithms that extract some metadata from the files and build the structures for the following information retrieval tasks performed by NPT-NH:

- run a Hidden Markov Model [11] speech recognition algorithm on media files for transcribing the speech to text;
- compute the TF-IDF measure for all articles and transcriptions, as specified in the CDI IDF algorithm [12];
- compute the page rank of articles, as specified in the weighted pagerank algorithm [13];
- perform a topic extraction routine on articles and transcriptions, as specified in the Latent Dirichlet Allocation algorithm [14].

These algorithms have to be implemented as described in the cited references and adapted to be run in the cluster managed by the parallel data processing framework.

4.5 Data Scaling and Maintenance

This section describes the processes performed for updating and scaling data in NPT-NH. Basically, these processes are executed integrally as batch tasks. New Pork Times' authors write new articles every day. Articles have to be placed in NPT-NH for allowing access to users. New authors may also come in. The following scaling tasks have to be performed by the system. New data has to be obtained as progressive slices of PRIMEBALL's corpus, making sure that

the data was not already inserted. Once new data are inserted, the system has to extract the necessary metadata from them so as to ensure that subsequent queries can be performed.

More specifically, when new information is inserted, it is necessary to recalculate some structures used for information retrieval. Thus, PRIMEBALL contains an implemented procedure for recomputing both TF-IDF metrics from all the documents and topics for all articles.

5 PRIMEBALL Workload

5.1 Query Set

This section contains the set of queries that are typically performed by NPT-NH users and that must be used for testing the performance of any given parallel data processing framework. These queries must be performed over the dataset defined in Section 4 and involve the most important aspects of performance. Although the following set of queries is not finite, it covers a wide range of classes from Figure 1, the relationships between them; it is applicable and enables to successfully measure performance.

One first subset of queries concerns the most common and hottest topics published in the system. The user might be interested in this information in order to know what kind of issues are the most frequently described in a certain interval of time, and therefore attract attention.

1. Articles containing the most frequent bigrams, sorted by pagerank.
2. Articles published during a time interval I, sorted by topics. The output contains pairs of article title and topic.
3. Most frequently used words by a journalist J from each country in the world during a certain time interval I. The output is compound by a list of countries, each one with keywords.
4. Rank of the keywords used in the articles published on an exact date D.
5. Most frequently used keywords in month M and year Y.

Furthermore, in order to evaluate how topics evolve w.r.t. time, it is necessary to include the various time measures in a second subset of queries.

6. Most frequently used keywords on day D and two different years Y_1 and Y_2, sorted by descending count.
7. Articles published on date D with the greatest number of references to the previous year.
8. Articles related to a topic T very frequently referenced lately, i.e., in a time interval I ending at the current date.
9. Journalists and professionals who wrote an article on the same month, on the same topic, sorted by days.

Finally, to analyze the diversity of articles and compare them w.r.t. their source, one should consider the following, eventual group of queries.

10. Articles written by X journalists in a specific time interval I that have at least Y common topics.
11. Rank of the languages in which articles are written. The output consists in pairs (language, number of articles).
12. Articles written by an author A from a given country C that best match a search term S.
13. Documents that best match another document published on the same day and month, but one year later.
14. Articles that focus on the same topic, but have been written by different journalists who were born before and after a given year Y.

5.2 Test Protocol

We present in this section different scenarios to help actually benchmark a system. To define them, a default scale factor SF is used. A dataset of scale factor SF is a set of articles and corresponding metadata having a total size of SF GB.

Scenario 1 This scenario simulates the evolution of the system along time, in terms of data operations and queries.

Initial state The system contains a dataset with a specified scale factor SF.

Operations

1. Execute queries 4, 7 and 14 from the generic query set (Section 5.1) choosing as date, e.g., September 12, 2001.
2. Double the volume of the dataset according to scenario 7.
3. Repeat the queries executed in step 1, for the same date and another one, e.g., November 5, 2008.

Scenario 2 This scenario simulates an extreme situation: a very famous article has been published with many mistakes and publishers are correcting it constantly. Their main concern is to deliver a consistent view of the article to people. Here, we are interested in measuring how many times the article is read in the older versions after being read once in the new version.

Initial state The system contains a dataset with a specified scale factor SF.

Operations

1. Initiate a thread performing 100 queries per second to retrieve the given article.
2. Start another thread updating the same article every 5 seconds.

Scenario 3 This scenario simulates node failures in terms of network reachability. Here, we are interested in knowing how many nodes can be removed from the cluster before some data become unreachable.

Initial state The system contains a dataset with a specified scale factor SF.

Operations

1. Execute all queries from the generic query set (Section 5.1) sequentially.
2. Remove a node and reiterate step 1.

Scenario 4 This scenario aims to measure the concurrency offered by the system while accessing data.

Initial state The system contains a dataset with a specified scale factor SF.

Operations Execute the following process 300 times and count the number of inconsistencies.

1. Start 10 threads executing the whole generic query set (Section 5.1).
2. Start 5 threads each of them performing:
   (a) updates in articles: 10 per second;
   (b) removing articles: 10 per second;
   (c) adding articles: 10 per second.

Scenario 5 The objective of this scenario is to simulate analysis procedures over the dataset.

Initial state The system contains a dataset with a specified scale factor SF.

Operations Execute queries uniformly selected from the following.

1. Top 10 articles seen each month in 2010.
2. Average number of pages per article for each journalist in the system.
3. Average age of publishers and standard deviation.
4. Maximum number of versions of an article.

Scenario 6 The aim of this scenario is to initialize the system and make it ready for handling the information for New Pork Times.

Initial state Empty storage system.

Operations Execute the steps described in Section 4.3 to initialize the environment.

Scenario 7 This scenario is used to increase the volume of the dataset to simulate the fact that new articles are inserted over time.

Initial state NPT-NH has a consistent state.

Operations Execute the steps described in Section 4.5 for data scaling and maintenance.

6 PRIMEBALL Properties and Metrics

6.1 Properties and Performance Metrics

This section presents the metrics that we use to evaluate the performance of the SUT. We also define the different system properties that can be assessed using PRIMEBALL.

We first specify two main metrics. The first one is throughput. The throughput of the SUT for a given scenario is the total time required to execute it (scenarios are defined in Section 5.2). The second metric, price performance, takes price (Section 6.2) into account and is expressed as follows.

$$Price\ performance = \frac{Throughput}{Price}$$

Moreover, the set of operations that can be executed against the system in the context of New Pork Times is defined as follows.

- Read: Obtain one or more articles.
- Write: Create a new article or a new version, add new journalists, languages, topics...
- Update: Modify an existing article within the same version or modify the information related to a journalist.
- Delete: Remove inappropriate content.
- Search: Obtain articles by matching a search (a set of given words, topics, authors, dates...).

Using all these definitions, we can set the following properties.

Generic Cloud Properties

1. **Scale up**: ability of the system to handle more data when adding more computers while maintaining performance.
   - *Importance*: In the case of a news website, it is very important to be able to scale up the system. There are a lot of news added every day and the service must keep on performing the same.

– *Measurement*: To measure this property, scenarios 4 and 5 (Section 5.2) must be executed twice, doubling the amount of data (SF) and the amount of nodes in the cluster the second time. Throughput increase ratio is the metric recommended for this property.

$$Throughput\ increase\ ratio = \frac{Throughput\ after}{Throughput\ before}$$

2. **Elastic speedup**: adding more computers to the cluster with the same amount of data results in better performance.
   – *Importance*: For New Pork Times, it is very relevant to know whether the system can offer a better performance when required, e.g., when there is a worldwide event with more people involved than usual looking for news and information. Thus, it is crucial to be able to maintain the quality of service even during peak demands.
   – *Measurement*: To measure this property, we propose to execute scenarios 2, 4 and 5 (Section 5.2) in order to observe throughput with the default cluster SF size. The metric we propose is also throughput increase ratio.

$$Throughput\ increase\ ratio = \frac{Throughput\ after}{Throughput\ before}$$

3. **Horizontal scalability**: ability of the system to distribute evenly the data load and workload among cluster nodes.
   – *Importance*: It is very useful to know up to what point one can exploit the current cluster and keep throughput in between some boundaries. In other terms, we determine what highest price performance can be achieved. It is very interesting in two senses:
     • upper bound: to answer the question "how many articles can the news website add into the system while keeping response time below 0.2 seconds";
     • lower bound: to optimize resource usage while fixing a performance lower bound. It might indeed be possible to reduce the number of nodes and offer the same user experience (response time).
   – *Measurement*: To assess this property, scenarios 4 and 5 (Section 5.2) must be executed and system throughput measured. Then, SF is increased and the process repeated. Again, throughput increase ratio can be used to evaluate this property. The closer it is to 1, the better is horizontal scalability.

$$Throughput\ increase\ ratio = \frac{Throughput\ after}{Throughput\ before}$$

4. **Latency**: time to execute a set of operations.
   – *Importance*: For New Pork Times, it is essential to be able to show news very quickly to users. If it takes too much time, users are going to look for a different website, thus a low latency is required.

- *Measurement*: Latency of the SUT can be measured as the throughput when executing scenarios 4, 5 and 6 (Section 5.2).

5. **Durability**: ability of the system to retain information for a long period of time.
   - *Importance*: In the case of a news website, it is very important to ensure that no information is lost. Users have to be able to check and find information they have read previously.
   - *Measurement*: Scenario 1 (Section 5.2) is intended to measure data durability. We define the durability ratio as a metric for this purpose.

$$Durability\ ratio = \frac{Correct\ reads}{Total\ reads}$$

6. **Consistency and version handling**: two different readings of the same data at the same time should return the same value.
   - *Importance*: It is important for a website to give a consistent view of data to all users at the same time around the world. In the proposed model, There may be several revisions of an article, which has to be consistent for all readers.
   - *Measurement*: Using scenario 2 (Section 5.2), the performance of the system for this property can be measured using the consistency ratio as a metric.

$$Consistency\ ratio = \frac{Consistent\ reads}{Total\ reads}$$

7. **Availability**: data is accessible even when there are some inaccessible nodes.
   - *Importance*: It is very relevant for New Pork Times to guarantee the access to all the news stored in the system.
   - *Measurement*: Scenarios 3 and 6 (Section 5.2) aim to measure this property, thus the throughput of the SUT can be taken as a metric for this property.

8. **Concurrency**: the system has to be able to offer a service to different clients at the same time.
   - *Importance*: In New Pork Times, users can keep reading while publishers are adding news, and the system has to be able to handle the multiple operations of different natures at the same time.
   - *Measurement*: Given concurrent scenario 4 (Section 5.2), we propose two metrics:
     - system throughput;
     - concurrency ratio.

$$Concurrency\ ratio = \frac{Successful\ operations}{Total\ operations}$$

Complex Data Properties

9. **Path traversals**: ability of the system to link data from different parts of the schema using the defined relationships.
   - *Importance*: In the case of a news Web site environment, this property is very important to improve search experience.
   - *Measurement*: Queries 3, 4, 7, 10 and 12 (Section 5.1) from the generic query set involve following a path through different class relations to link concepts. The throughput of this type of queries is used to measure that property.
10. **Construction of complex results**: ability of the system to generate (semi)-structured output from the information system.
    - *Importance*: This property is very relevant to a news website, mainly to allow analysis over the contained data.
    - *Measurement*: The generic query set defined in Section 5.1 contains queries with complex results, i.e., queries 2, 3, 6 and 11. The throughput of these queries can be used as a metric for this property. Moreover, scenario 6 (Section 5.2) has to be used to measure this property.
11. **Polymorphism**: ability of the system to deal with type inheritances, i.e., treating types and subtypes of objects to compute query results.
    - *Importance*: Inheritance is a good way to deal with complex relationships between objects. For this reason, the performance of the system while executing these kinds of operations is very relevant.
    - *Measurement*: Fix a cluster and an initial workload, then execute and measure system performance while executing queries 3, 9, 12 and 15 (Section 5.1; all of them involve inheritance operations).

Big Data Properties

12. **Analysis**: ability of the system to generate summarized data and statistical information.
    - *Importance*: For a news website, having statistics such as how many times an article has been read, average words per article, etc., is very relevant.
    - *Measurement*: This property can be measured in terms of throughput while executing analytical scenario number 5 (Section 5.2).

Information Retrieval Properties

13. **Full text**: being able to search a single word in all documents simultaneously.
    - *Importance*: This property is very relevant to a news website to allow users searching information easily in the system.
    - *Measurement*: It can be measured in terms of throughput when searching for different terms, some famous, some normal and some strange, e.g., Obama, Higgs, Star Trek, Cleopatra, etc. Queries of this type are included in the generic query set (Section 5.1).

6.2 Pricing

In Section 6.1, we defined sytem performance w.r.t. time and cost. The main pricing factors involved in processing data in the cloud follow.

- Cloud provider: different cloud service providers may have different pricing policies.
- Number of instances and type: infrastructure used to execute PRIMEBALL.
- Required storage space: it is directly related to the scale factor used (Section 5.2).
- Platform inherent costs: operation, administration and maintenance.
- Execution time: time spent to run the tests.

The specific cloud provider model has to be applied to compute real cost.

7 Conclusions

We propose in this paper the specifications of PRIMEBALL, a complete and unified benchmark for measuring the characteristics of parallel cloud processing frameworks for big data applications. The real-life model adopted in PRIME-BALL is that of a fictitious news hub called New Pork Times, which is basically a fair approximation of a popular real-life news site. The general architecture and inner processes of the system are well-defined so as to allow an unambiguous implementation of the benchmark.

The workload applied on this news dataset is not only made of queries, but also of data-intensive batch processes. Moreover, we propose several use-case scenarios together with relevant metrics for assessing the framework's performance from different points of view, such as data availability or horizontal scalability. The novelty of our work lies in the fact that existing, related benchmarks measure parallelization capabilities, cloud features, big data analysis ability, but none of them combines all these properties while exploiting real-life data.

Future work on PRIMEBALL will be mainly focused on implementing a crawler for fetching and transforming real data from the Web to feed the benchmark's dataset. Moreover, the actual feasibility and relevance of PRIMBALL shall be validated by implementing the benchmark in several cloud environments and by publishing performance comparison results. For instance, implementation in popular data processing frameworks such as Hadoop should be achieved.

Moreover, future extensions of the benchmark could include new scenarios that exploit different properties of cloud providers, such as vertical growth of the cluster, or new measures such as efficiency of bandwidth use. Actual experiments should also help refine the benchmark's workload.

References

1. IBM, What is big data? http://www-01.ibm.com/software/data/bigdata/, 2012.
2. K. Sato, An Inside Look at Google BigQuery, White paper, https://cloud.google.com/files/BigQueryTechnicalWP.pdf, 2012.
3. K. Shvachko, H. Kuang, S. Radia, R. Chansler, The Hadoop Distributed File System, 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST 2010), Incline Village, USA, pp. 1-10, 2010.
4. E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, C. Tosun, Benchmarking in the Cloud: What It Should, Can, and Cannot Be, 4th TPC Technology Conference (TPCTC 2012), Istanbul, Turkey; LNCS 7755, pp. 173-188, 2013.
5. Transaction Processing Performance Council (TPC), TPC Benchmark DS Standard Specification Version 1.1.0, http://www.tpc.org, 2012.
6. Open Cloud Consortium, Generate synthetic site-entity log data for testing and benchmarking applications requiring large data sets, http://code.google.com/p/malgen/, 2009.
7. Cloud Harmony, http://www.cloudharmony.com/benchmarks, 2013.
8. B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, S. Sears, Benchmarking cloud serving systems with YCSB, 1st ACM symposium on Cloud Computing (SoCC 2010), Indianapolis, USA, pp. 143-154, 2010.
9. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: amazon's highly available key-value store, 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP 2007), pp. 205-220, 2007.
10. Y. Chen, S. Alspaugh, A. Ganapathi, R. Griffith, R. KatzThe, Statistical Workload Injector for MapReduce (SWIM), https://github.com/SWIMProjectUCB/SWIM/wiki, 2013.
11. B.H. Juang, L.R. Rabiner, Hidden Markov models for speech recognition, Technometrics 33(3), pp. 251-272, 1991.
12. M. Xu, H. Liang, L. Xin, A Refined TF-IDF Algorithm Based on Channel Distribution Information for Web News Feature Extraction, Second International Workshop on Education Technology and Computer Science (ETCS 2010), Wuhan, China, Vol. 2, pp. 15 - 19, 2010.
13. W. Wing, A.A. Ghorbani, Weighted pagerank algorithm, Second Annual Conference on Communication Networks and Services Research (CNSR 2004), Fredericton, Canada, pp. 305-314, 2004.
14. D. Newman, A. Asuncion, P. Smyth, M. Welling, Distributed algorithms for topic models, The Journal of Machine Learning Research 10, pp. 1801-1828, 2009.