MRBS: A Comprehensive MapReduce Benchmark Suite

Amit Sangroya
INRIA - LIG
Grenoble, France
Amit.Sangroya@inria.fr

Damián Serrano
INRIA - LIG
Grenoble, France
Damian.Serrano@inria.fr

Sara Bouchenak
University of Grenoble - LIG - INRIA
Grenoble, France
Sara.Bouchenak@inria.fr

***Abstract*—MapReduce is a promising programming model for distributed data processing. Extensive research has been conducted on the scalability of MapReduce, and several systems have been proposed in the literature, ranging from job scheduling to data placement and replication. However, realistic benchmarks are still missing to analyze and compare the effectiveness of these proposals. To date, most MapReduce techniques have been evaluated using microbenchmarks in an overly simplified setting, which may not be representative of real-world applications. This paper presents MRBS, a comprehensive benchmark suite for evaluating the performance of MapReduce systems. MRBS includes five benchmarks covering several application domains and a wide range of execution scenarios such as data-intensive vs. compute-intensive applications, or batch applications vs. online interactive applications. MRBS allows to characterize application workload and dataload, and produces extensive high-level and low-level performance statistics. We illustrate the use of MRBS with Hadoop clusters running on Amazon EC2.**

***Keywords*-Benchmark; Performance; MapReduce; Hadoop; Cloud Computing**

I. Introduction

MapReduce has become a popular programming model and runtime environment for developing and executing distributed data-intensive and compute-intensive applications [1]. It offers developers a means to transparently handle data partitioning, replication, task scheduling and fault tolerance on a cluster of commodity computers. MapReduce allows a wide range of applications such as log analysis, data mining, web search engines, scientific computing [2], bioinformatics [3], decision support and business intelligence [4].

There has been a large amount of work on MapReduce towards improving its performance and scalability. Several efforts have explored task scheduling policies in MapReduce [5], [6], [7], cost-based optimization techniques [8], resource provisioning [9], replication and partitioning policies [10], [11]. There has also been a considerable interest in extending MapReduce with techniques from database systems [12], [13], [14], [15].However, there has been very little in the way of empiric evaluation of the performance of these systems. Most evaluations have relied on microbenchmarks based on simple MapReduce programs. While microbenchmarks may be useful in targeting specific system features, they are not representative of full distributed applications, and they do not provide multi-user realistic workloads. Thus, for a benchmark suite to enable a thorough analysis of MapReduce systems, it must provide the following. First, it must enable the *empirical evaluation of the performance* of MapReduce systems. This will provide a means to analyze the effectiveness of scalability , a key feature of MapReduce. Furthermore, with the advent of MapReduce-enabled cloud environments and the pay-as-you-go model, a benchmark suite must allow the *evaluation of the costs* of MapReduce systems in cloud environments. Second, it must cover a *variety of application domains and workload characteristics*, ranging from compute-oriented to data-oriented applications, batch applications to online real-time applications. Indeed, while MapReduce frameworks were originally limited to offline batch applications, recent works are exploring the extension of MapReduce beyond batch processing [16], [17]. Thus, a benchmark suite must consider different execution modes. Moreover, in order to stress MapReduce performance, the benchmark suite must enable different workload injection profiles and concurrency levels. Finally, the benchmark suite must be *portable and easy to use on a wide range of platforms* and cloud infrastructures.

More specifically, the contributions of the paper are as follows:

- The paper presents *MapReduce Benchmark (MRBS)*, a comprehensive benchmark suite for evaluating the performance of MapReduce systems. MRBS covers five application domains: recommendation systems, business intelligence and decision support systems, bioinformatics, text processing, and data mining. MRBS provides a total of 32 different request types implemented as MapReduce programs, and 26 sets of configurations and input data. The workload profile can be set by the user of MRBS to represent different scenarios.
- The paper provides performance characterizations of MRBS benchmarks through extensive experimental evaluations. It provides measures in the form of request response time, throughput, and cost. It also describes the benchmarks data access and processing patterns, and provides low-level MapReduce statistics such as the throughput of MapReduce jobs and tasks, the throughput of reads and writes.
- The paper illustrates the use of MRBS with Hadoop

MapReduce clusters running on Amazon EC2.It also presents two case studies of MRBS to investigate the scalability of Hadoop MapReduce.

MRBS prototype currently consists of 10 Klines of Java source code. MRBS is easy to use, and allows automatic deployment of experiments to cloud infrastructures. It does not depend on any particular infrastructure and can run on different private or public clouds, such as Amazon EC2. Integrating a new cloud infrastructure into MRBS requires only few lines of code (approximately 350 lines). MRBS is available as a software framework to help researchers and practitioners to better analyze and evaluate the performance and scalability of MapReduce systems in different settings and applications domains.

The remainder of the paper is organized as follows. Section II describes the background on MapReduce. Sections III-V describe the MRBS benchmark suite, its architecture and design principles. Section VI presents experimental results, and Section VII discusses use cases of MRBS. Section VIII reviews the related work, and Section IX draws our conclusions.

II. System and Programming Model

MapReduce is a programming model and a software framework introduced by Google in 2004 to support distributed computing and large data processing on clusters of commodity machines [1]. They are achieved by automatic task scheduling in MapReduce clusters, automatic data placement, partitioning and replication, and automatic failure detection and task re-execution. MapReduce successfully supports a wide range of applications such as image analytics, next-generation sequencing, recommendation systems, search engines, social networks, business intelligence, and log analysis.

MapReduce is a functional programming model that provides a simple means to write programs that process large input data sets. Programmers write only two main functions: a *map* function and a *reduce* function, and the MapReduce framework automatically handles data and computation distribution in a cluster. A MapReduce *job*, i.e. an instance of a running MapReduce program, has several phases; each phase consists of multiple *tasks* scheduled by the MapReduce framework to run in parallel on cluster nodes. First, input data are divided into *splits*, one split is assigned to each map task. During the mapping phase, tasks execute a *map* function to process the assigned splits and generate intermediate output data. Intermediate outputs are grouped into distinct subsets called *partitions*; these partitions are used as inputs of reduce tasks. Then, the reducing phase runs tasks that execute a *reduce* function to process intermediate data and produce output data.

There are many implementations of MapReduce among which the popular open-source Hadoop framework [18]. Hadoop is also available in public clouds such as Amazon EC2 [19], Google App Engine [20] or Open Cirrus [21]. A Hadoop cluster consists of a *master node* and *slave nodes*. Users (i.e. clients) of a Hadoop cluster submit MapReduce jobs to the master node which hosts the *JobTracker* daemon that is responsible of scheduling the jobs. By default, jobs are scheduled in FIFO mode and each job uses the whole cluster until the job completes. However, other multi-user job scheduling approaches are also available in Hadoop to allow jobs to run concurrently on the same cluster. This is the case of the *fair scheduler* which assigns every job a fair share of the cluster capacity over time. Job scheduling policy can be set in *mapred.jobtracker.taskScheduler* Hadoop property. Moreover, each slave node hosts a *TaskTracker* daemon that periodically communicates with the master node to indicate whether the slave is ready to run new tasks. If it is, the master schedules appropriate tasks on the slave.

Hadoop framework also provides a distributed filesystem (HDFS) that stores data across cluster nodes [22]. HDFS architecture consists of a *NameNode* and *DataNodes*. The *NameNode* daemon runs on the master node and is responsible of managing the filesystem namespace and regulating access to files. A *DataNodes* daemon runs on a slave node and is responsible of managing storage attached to that node. HDFS is thus a means to store input, intermediate and ouput data of Hadoop MapReduce jobs.

III. Design Philosophy

The motivation of this work is to come up with a comprehensive benchmark suite for MapReduce systems. More precisely, the objectives of the MRBS Benchmark suite are as follows:

1) **Multi-criteria analysis.** MRBS aims to measure and analyze multiple aspects of the performance of MapReduce systems. In particular, we consider several measurement metrics such as client request latency, throughput, and cost. We also consider low-level MapReduce metrics, such as size of read/written data, throughput of MapReduce jobs and tasks, etc.
2) **Diversity.** MRBS covers a variety of application domains and programs with a wide range of MapReduce characteristics. This includes data-oriented applications vs. compute-oriented applications. Furthermore, whereas MapReduce was originally used for long running batch jobs, another use case has emerged where a MapReduce cluster is shared between multiple users running interactive requests over a common data set. Therefore, MRBS considers batch applications as well as interactive applications. Moreover, MRBS allows to characterize different aspects of application load such as the *workload* and the *dataload*. Roughly speaking, the workload is characterized by the number of clients (i.e. users) sharing a MapReduce cluster and the types of client requests (i.e. MapReduce programs). And the

dataload characterizes the size of MapReduce input data.

3) **Usability.** MRBS is easy to use, configure and deploy on a MapReduce cluster. It is independent from any infrastructure and can easily run on different public clouds and private clouds. MRBS provides results which can be readily interpreted in the form of monitored statistics and automatically generated charts.

IV. System Architecture

The architecture of the MRBS framework is described in Figure 1. It consists of four main elements: the set of benchmarks, the cluster configuration component, the load injector, and the statistics monitoring component. These elements are presented in the following.

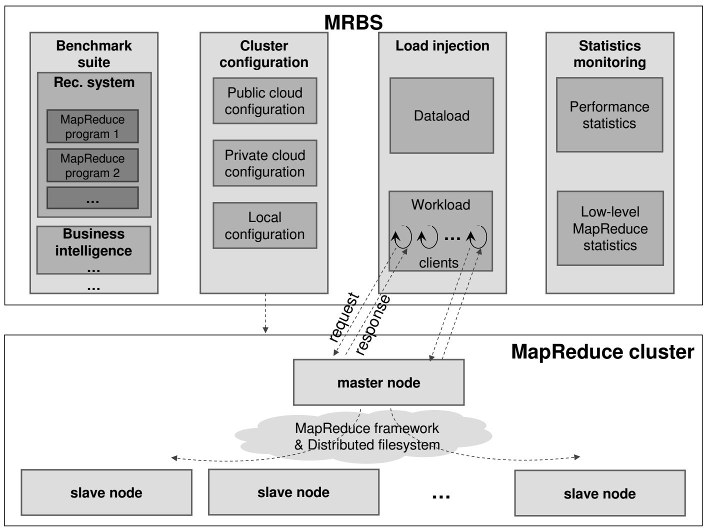


Figure 1. MRBS architecture

A. Benchmark Suite

Conceptually, a benchmark in MRBS implements a service that provides different types of operations, which are requested by clients. The benchmark service is implemented as a set of MapReduce programs running on a cluster, and clients are implemented as external entities that remotely request the service. Depending on the complexity of a client request, the request may consist of one or multiple successive MapReduce jobs. A benchmark has two execution modes: interactive mode or batch mode. In interactive mode, a client interacts with the service in a closed loop where he/she requests an operation, waits for the request to be processed, receives a response, waits a think-time, before requesting another operation. In batch mode, the client and benchmark interactions are modeled as an open loop.

A benchmark run has three successive phases: a warm-up phase, a run-time phase, and a slow-down phase, which length may be chosen by the end-user of the benchmark. The end-user may also choose the number of times a benchmark is run, to produce average statistics.

MRBS benchmark suite consists of five benchmarks covering various application domains such as recommendation systems, business intelligence, bioinformatics, text processing, and data mining. The user can choose the actual benchmark. The benchmarks and the types of operations they provide are detailed in Section V.

B. Cluster Configuration

MRBS *cluster configuration component* is responsible of setting up a cluster on which the benchmark will run. The infrastructure hosting the cluster and the size of the cluster are configuration parameters of MRBS and can be given different values. MRBS acquires on-demand resources provided by cloud computing infrastructures such as private clouds, or Amazon EC2 public cloud [23], and automatically releases the resources when the benchmark terminates. We expect to provide MRBS versions for other cloud infrastructures such as OpenStack open source cloud infrastructure [24]. Once the cluster is set up, the MapReduce framework and its underlying distributed file system are started. The current implementation of MRBS uses the popular Apache Hadoop MapReduce framework and HDFS distributed file system.

C. Load Injection

Once the MapReduce cluster is set up, it is ready to run MapReduce jobs following a given load. Interestingly, our benchmark framework considers different aspects of load: dataload and workload. This allows an MRBS end-user to easily define different scenarios for a target benchmark, and to stress different the performance and scalability of MapReduce systems.

The *dataload* is characterized by the size and nature of data sets used as inputs for a benchmark. Obviously, the nature and format of data depend on the actual benchmark and its associated MapReduce programs. For instance, the MRBS movie recommendation system benchmark takes input data consisting of users, movies and ratings users give to movies. Whereas the bioinformatics benchmark uses input data in the form of genomes for DNA sequencing.

The *workload* of a benchmark is first characterized by the number of concurrent clients. It is also characterized by client request distribution, that is the relative frequencies of different request types. Request distribution may be defined using a state-transition matrix that gives the probability of transitioning from one request type to another. Request transitions may also follow other distribution laws such as a random distribution.

Finally, once a workload and a dataload are defined, MRBS injects that load into the MapReduce cluster. It first uploads input data in the MapReduce distributed file system. This is done once, at the beginning of the benchmark, and the data are then shared by all client requests. It then creates as many threads as concurrent clients there are. Thread clients will remotely send requests to the master node of the MapReduce cluster which schedules MapReduce jobs in the cluster. If parameters are associated with client requests, their values are randomly generated.

D. Using MRBS

MRBS comes with a configuration file that involves several parameters among which the following: the actual benchmark to use, the length of the benchmark warm-up phase, runtime phase, and slow-down phase, the size of the benchmark input data set, the size of the MapReduce cluster, the cloud infrastructure that will host the cluster, in addition to workload characteristics described in the previous section (e.g. number of concurrent clients , etc.). To keep the use of MRBS simple, these parameters have default values that may be adjusted by MRBS user.

MRBS produces various runtime performance statistics. These include client request response time, throughput, and financial cost. Statistics are provided in the form of average values, as well as detailed values for each client request type. MRBS also provides low-level MapReduce statistics related to the number, length of MapReduce jobs, map tasks, reduce tasks, the size of data read from or written to the distributed file system, etc. These low-level statistics are built using Hadoop counters. Optionally, MRBS can generate charts plotting continuous-time results. It can also perform multiple runs of the same benchmark configuration in order to report average statistics.

V. Benchmark Suite

Table I briefly describes MRBS benchmark suite which covers five application domains. The upper part of the table shows real-world applications, and the lower part of the table represent collections of diverse MapReduce programs. The benchmark suite provides a total of 32 different types of client requests implemented as MapReduce programs. It allows 26 different combinations of execution modes and input data. Workload is another configuration parameter that can be set by the user of MRBS to represent different scenarios.

The benchmarks exhibit different behaviors in terms of computation pattern and data access pattern: the recommendation system is a compute-intensive benchmark, the business intelligence system is a data-intensive benchmark, and the other benchmarks are relatively less compute/data-intensive. This is shown in Figure 2 that compares the different benchmarks (note the logarithmic scale). Figure 2(a) gives the average size of data accessed per client request, and Figure 2(b) gives the ratio of request processing time to the size of accessed data. Moreover, Figure 3 shows that MRBS benchmarks present different MapReduce characteristics in terms of the average number of MapReduce jobs and tasks per client request. In the following, we detail MRBS benchmarks.

A. Recommendation System

Recommendation systems are widely used in e-commerce sites such as Amazon.com which, based on purchases and site activity, recommends books likely to be of interest. MRBS implements an online movie recommender system. It builds upon a set of movies, a set of users, and a set of ratings and reviews users give for movies to indicate whether and how much they liked or disliked the movies. These data have been collected from a real movie recommendation web site [25]. The benchmark provides four types of operations. First, a user may ask for all ratings and reviews given by other users for a given movie, to see whether people liked or disliked the movie. The recommendation system also allows to browse all ratings and reviews given by a user. Furthermore, a user may ask the recommender system to provide him/her the top ten recommendations, these are the movies this user would like the most. Another type of operation a user may perform is to ask the recommendation system how it would recommend him/her a given movie; this would indicate to the user whether and how much he/she would like or dislike that movie.

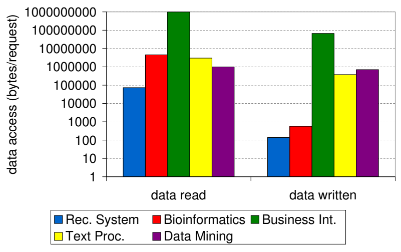
This benchmark is based on MapReduce implementations of data mining and search algorithms. For building recommendations, similarities between movies are computed by looking to users' ratings and preferences. The algorithm uses item-based recommendation techniques to find movies that are similar to other movies [26]. Similarities between movies are relatively static and can thus be computed once and then reused. This is what the Recommendation system benchmark does by storing the precomputed similarities in the distributed filesystem. Therefore, the benchmark handles client requests by applying a search algorithm on the precomputed data.

B. Business Intelligence

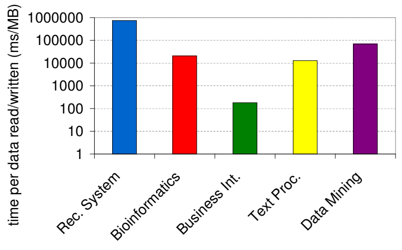
The Business Intelligence benchmark represents a decision support system for a wholesale supplier. It implements business-oriented queries that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions. MRBS includes a MapReduce implementation of the TPC-H industry-standard benchmark [27]. It uses Apache Hive on top of Hadoop, a data warehouse that facilitates ad-hoc queries using a SQL-like language called HiveQL [28]. The benchmark consists of eight data tables, and provides 22 types of operations among which an operation that identifies geographies where there are customers who may be likely to make a purchase, or an operation that retrieves the ten unshipped orders with the highest value. The provided operations are implemented

Table I
APPLICATION DOMAINS AND BENCHMARK CHARACTERISTICS IN MRBS. AN APPENDED '+' SYMBOL INDICATES HIGHER DATALOAD, HIGHER COMPUTATION CONTENTION OR HIGHER DATA ACCESS CONTENTION. AN APPENDED '*' SYMBOL INDICATES A RECOMMENDED CONFIGURATION.

Domain	**Dataload**	**Execution mode**	**Workload**	**Computation vs. data access**
Recommendation system	dataload 100,000 ratings, 1000 users, 1700 movies dataload+ 1 million ratings, 6000 users, 4000 movies dataload++ 10 million ratings, 72,000 , 10,000 movies	interactive* / batch	mono-user / multi-user	compute-oriented+
Business intelligence	dataload 1GB dataload+ 10GB dataload++ 100GB	interactive* / batch	mono-user / multi-user	data-oriented+
Bioinformatics	dataload genomes of 2,000,000 to 3,000,000 DNA characters	interactive* / batch	mono-user / multi-user	data-oriented / compute-oriented
Text processing	dataload text files (1GB) dataload+ text files (10GB) dataload++ text files (100GB)	interactive / batch*	mono-user / multi-user	data-oriented / compute-oriented
Data mining	dataload 5000 documents, 5 newsgroups, 600 control charts dataload+ 10,000 documents, 10 newsgroups, 1200 control charts dataload++ 20,000 documents, 20 newsgroups, 2400 control charts	interactive / batch*	mono-user / multi-user	data-oriented / compute-oriented



(a) Data access per client request



(b) Processing time per data size

Figure 2. Data-oriented vs. compute-oriented benchmarks[1]

as HiveQL queries, that are translated into MapReduce programs by the Hive framework. The input data of the benchmark were generated with the DBGen TPC provided software package, and are compliant with the TPC-H specification [27].

C. Bioinformatics

The Bioinformatics benchmark performs DNA sequencing. Users of the benchmark may choose a complete genome to analyze among a set of genomes. Roughly speaking, DNA sequencing attempts to find where reference reads (i.e. short DNA sequences) occur in a genome, allowing a fixed number of errors. This is a highly parallelizable process that can benefit from MapReduce. The benchmark includes a MapReduce-based implementation of DNA sequencing [3]. The data used in the benchmark are publicly available genomes [29]. Currently, the benchmark allows to analyze several genomes of organisms such as the pathogenic organisms Salmonella Typhi, Rhodococcus equi, and Streptococcus suis. The Salmonella Typhi is a parasite that causes human typhoid fever; its genome is about 2,000,000 DNA characters long. The Rhodococcus equi is a disease-causing organism in horses, with a genome of about 3,000,000 DNA characters. Streptococcus suis is another pathogen which human infection can cause severe outcomes such as meningitis; its genome is also about 2,000,000 DNA characters long. The benchmark can be easily extended with new genomes to analyze by simply defining them as new input data in MRBS configuration file.

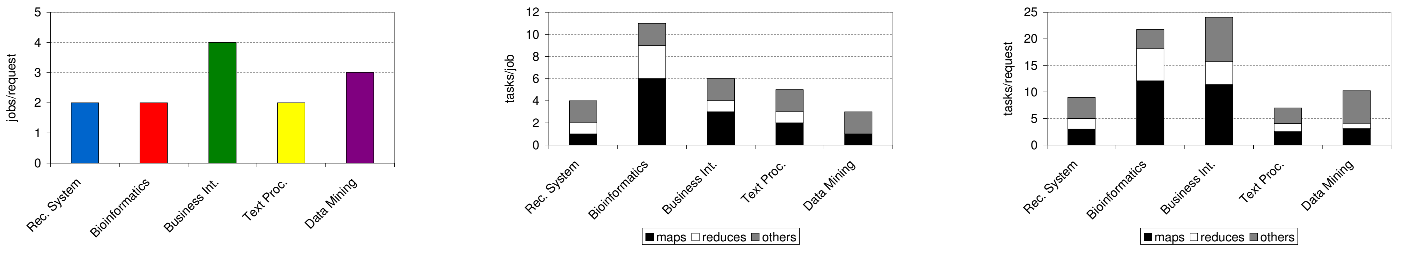
D. Text Processing

Text processing is a classical application of MapReduce used, for instance, to analyze the logs of web sites and search engines. MRBS provides a MapReduce text processing-oriented benchmark, with three types of operations allowing clients to search words or word patterns in text documents, to know how often words occur in text documents, or to sort the contents of documents. The benchmark uses synthetic input data that consist of randomly generated text files of different sizes.

E. Data Mining

This benchmark provides two types of data mining operations: clustering and classification [30]. Bayesian classification assigns a category from a fixed set of known

[1] Figures obtained with the first dataload of each benchmark, running on a ten-node Hadoop cluster, with ten concurrent clients.



(a) MapReduce jobs per client request

(b) MapReduce tasks per job

(c) MapReduce tasks per client request

Figure 3. MapReduce characteristics of benchmarks

categories to an un-categorized element. As an example of classification application, Yahoo! Mail classifies incoming messages as spam, or not, based on prior emails and spam reports. MRBS benchmark considers the case of classifying newsgroup documents into categories. A first step consists in applying a learning algorithm to train the model. Then, the model can be used with an un-classified document to estimate the newsgroup the document is likely to belong to. The benchmark uses collections of data publicly available from [31].

Furthermore, the benchmark provides canopy clustering operations. Canopy clustering partitions a large number of elements into clusters in such a way that elements belonging to the same cluster share some similarity. For instance, Google News uses clustering techniques to group news articles according to their topic. The benchmark uses datasets of synthetically generated control charts, to cluster the charts into different classes based on their characteristics [30].

VI. Evaluation

A. Experimental Setup

The experiments presented the following were conducted on Amazon EC2 cloud computing infrastructure [23]. Each execution of MRBS was performed on a set of Amazon EC2 instances (i.e. virtual nodes), that consists of one node hosting MRBS and emulating clients, and a cluster of nodes hosting the MapReduce distributed framework. The MapReduce cluster runs on Amazon EC2 large instances, that are 64-bit servers, providing 4 EC2 Compute Units in two virtual cores and equipped with 7.5 GB of RAM and 850MB of data storage. One EC2 Compute Unit is equivalent to the CPU capacity of a 1.0-1.2 GHz 2007 Opteron processor. The hourly price of a large instance is $0.34.

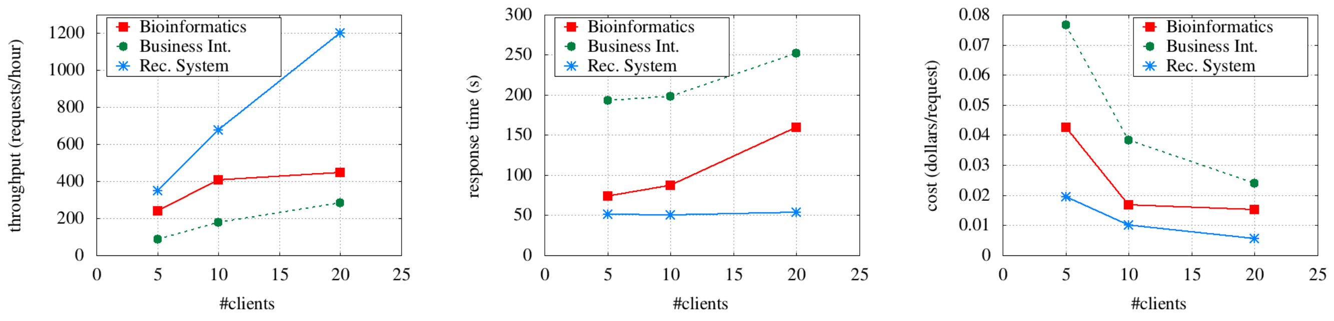
The software environment used in the experiments is as follows. Amazon EC2 instances run Fedora Linux 8 with kernel v2.6.21. The MapReduce framework is Apache Hadoop v0.20.2 [18], and Hive v0.7 [28], on Java 6. MRBS uses Apache Mahout v0.6 data mining library[30], and CloudBurst v1.1.0 DNA sequencing library [3]. The following experiments with MRBS consider a workload where client requests follow a random distribution. The results presented in the following correspond to the average of three executions of 30 minutes run-time, after a 15 minutes of warm-up. We conducted a set of experiments on Amazon EC2 public cloud and on a private cloud, with all MRBS benchmarks. However, due to space limitation, the paper only presents the results of experiments on Amazon EC2 with the recommendation system, business intelligence and bioinformatics benchmarks, three benchmarks representing real-world applications.

B. Performance Evaluation

To illustrate how MRBS evaluates the performance of MapReduce systems, we present here some experimental results. In these experiments, we vary the workload to see how this impacts performance. Figure 4 shows the performance statistics obtained with a 20 node Hadoop cluster when the number of concurrent clients increases. Each benchmark uses its first dataload configuration as input data (see Table I).

Figure 4(a) presents the throughput of each benchmark, that is the number of clients requests handled by the benchmark per unit of time. Figure 4(b) shows the average client response time. Response time is the elapsed time from the moment the client submits a request until the response is received by the client. This may include, not only the execution time of that request, but also the overhead of time-sharing in Hadoop cluster.

The throughput of the Recommendation System increases quasi-linearly with the number of clients. This is due to the fact that the MapReduce system is not overloaded and can thus cope with more concurrent clients. This is also confirmed by Figure 4(b) that shows that the response time of the Recommendation System does not increase with the number of concurrent clients. The throughput of the Bioinformatics and Business Intelligence benchmarks increases



(a) Throughput (b) Client request response time (c) Cost

Figure 4. Performance under different workloads

linearly between 5 and 10 clients. The throughput with 20 clients slightly increases for Business Intelligence, whereas it does not increase appreciably with Bioinformatics, which reaches its maximum capacity. This is also reflected in the response times of the Bioinformatics and Business Intelligence benchmarks, which present a sharp increase between 10 and 20 clients.

Thus, these experiments show that a MapReduce cluster is able to successfully host multi-user applications. Interestingly, MRBS benchmarks show different throughput speedups, and this is explained as follows. The lower the average number of MapReduce tasks per job in a benchmark is, the lower the number of task slots needed by the benchmark to run a request in the MapReduce cluster is, thus, the higher the number of available task slots for other concurrent client requests is, and the higher the benchmark throughput is. This is confirmed by the results shown in Figures 3(b) and 4(a). Consequently, thanks to concurrency in the MapReduce cluster, the average cost of a client request is reduced by a factor of up to 2-3, depending on the benchmark, cf. Figure 4(c).

Furthermore, MRBS also produces low-level MapReduce performance statistics, in terms of the number of MapReduce tasks per unit of time, and the size of data read or written in the distributed filesystem per unit of time, as respectively shown in Figures 5(a), 5(b) and 5(c). More specifically, the Business Intelligence benchmark presents the highest amount of data read/written. For reads, it is two orders of magnitude higher than Bioinformatics, and three orders higher than the Recommendation System. For writes, both Bioinformatics and Recommendation System write few data compared to the Business Intelligence benchmark, which is five orders of magnitude higher. This corroborates the results of client request response times, the Business Intelligence benchmark being the one with the highest response time.

VII. Case Studies

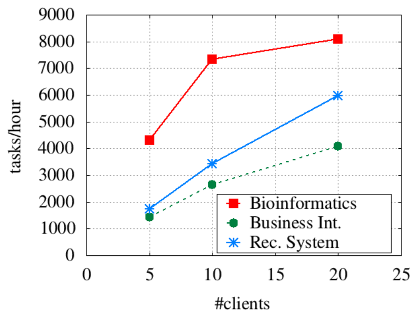
MRBS can be used for many purposes, such as comparing different hardware configurations for running a MapReduce system, evaluating data placement strategies, or cost-based optimization techniques. To illustrate the use of MRBS, we present two case studies where we investigate the scalability of MapReduce systems with regard, on the one hand, to the size of MapReduce clusters, and on the other hand, to the size of MapReduce input data.

A. *Scalability With Regard To Cluster Size*

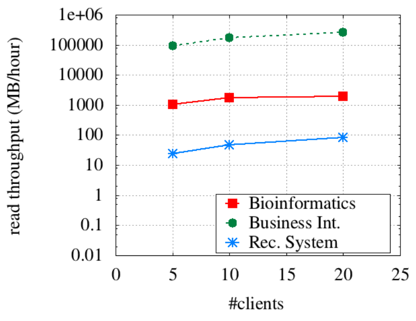
In this case study, we evaluate the scalability of Hadoop MapReduce with regard to the size of Hadoop clusters. We conducted experiments with MRBS running on Hadoop clusters of different sizes: 5, 10, 20, and 25 nodes. We compare the results of theses clusters with the results obtained when running MRBS on one node. Figure 6 presents the performance results of the experiments, with the Bioinformatics, Business Intelligence and Recommendation System benchmarks, running 10 concurrent clients. Figures 6(a) and 6(b) respectively show the response time speedup and throughput speedup as functions of cluster size. The higher the response time speedup is, the better (i.e. lower) the response time is. Similarly, the higher the throughput speedup is, the better (i.e. higher) the throughput is.

Here, response time results show that, up to 5 nodes, the Hadoop cluster is able to scale linearly when it runs Bioinformatics or Business Intelligence applications. With higher cluster sizes, the speedup is sublinear. The three benchmarks achieve the maximum speedup with respectively 10 nodes for the Recommendation System, 20 nodes for the Bioinformatics benchmark, and 25 nodes for the Business Intelligence system. These differences in scalability capabilities are explained by the fact that Bioinformatics and Business Intelligence benchmarks have a much higher number of MapReduce tasks per client request than the Recommendation System (cf. Figure 3(c)). Thus, the two former benchmarks are able to exploit concurrency when having more nodes.

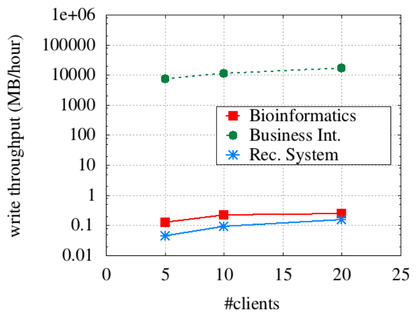
Figure 6(c) describes the average cost of a client request as a function of the number of nodes in Hadoop clusters. Obviously, the larger is the Hadoop cluster running on



(a) MapReduce task throughput

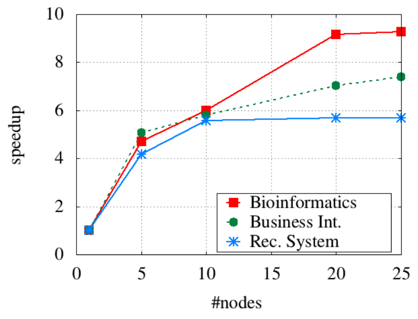


(b) DFS read throughput

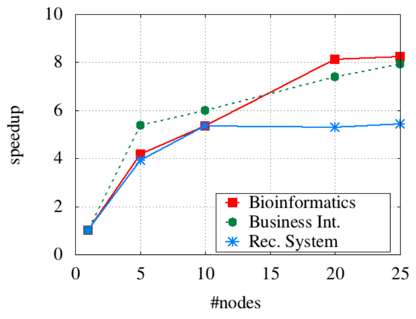


(c) DFS write throughput

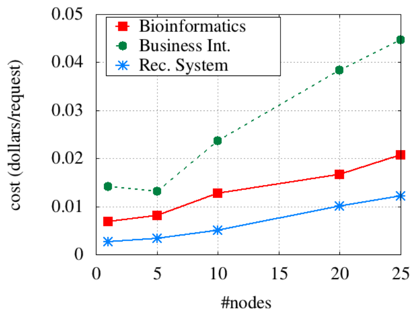
Figure 5. Low-level MapReduce statistics



(a) Response time speedup



(b) Throughput speedup



(c) Cost

Figure 6. Performance under different cluster scales

Amazon EC2, the higher is the cost of a client request. Surprisingly, the Business Intelligence benchmark shows that a five node Hadoop cluster costs less than one node. This is explained by the fact that with 5 nodes, Business Intelligence achieves a superlinear speedup in throughput, and request cost is a function of throughput and Amazon EC2 hourly cost model.

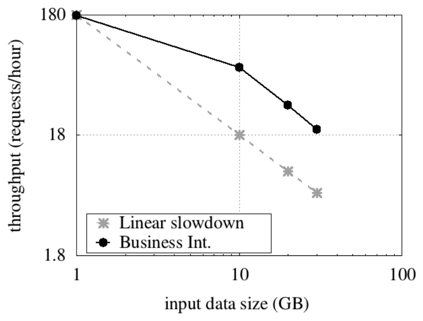
B. *Scalability With Regard To Data Size*

In the second case study, we investigate the scalability of Hadoop MapReduce with respect to the size of input data. We conducted experiments with the MRBS Business Intelligence benchmark, with different sizes of input data: 1 GB, 10 GB, 20 GB, and 30 GB. The experiments were conducted on a 20 node Hadoop cluster with 10 concurrent clients. Figure 7 presents performance results as functions of input data size. The measured performance of the Business Intelligence benchmark is compared with the theoretical linear performance slowdown that we could expect when increasing the data size.

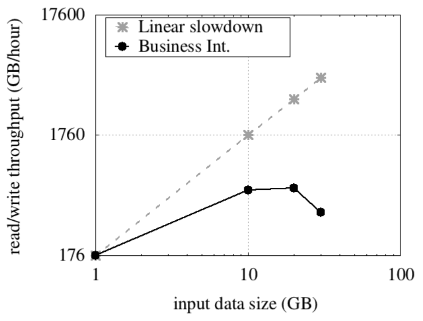
Figure 7(a) shows that, even though the performance decreases when the input data are larger, request throughput performs better than linear slowdown (note the logarithmic scale of the figure). Here, throughput is three times better than linear slowdown. This is due to the fact that accesses to the Hadoop filesystem do not increase linearly with the size of input data, as shown in Figure 7(b). Figure 7(b) describes the throughput of data read or written in the Hadoop filesystem, and Figure 7(c) describes MapReduce task throughput. The results show that with input data larger than 10 GB, the Hadoop cluster is overloaded, since it executes more MapReduce tasks while handling less client requests. This is also confirmed by other statistics reported by MRBS and showing an increase of MapReduce task retries, up to +25%, when input data are larger than 10 GB.

VIII. Related Work

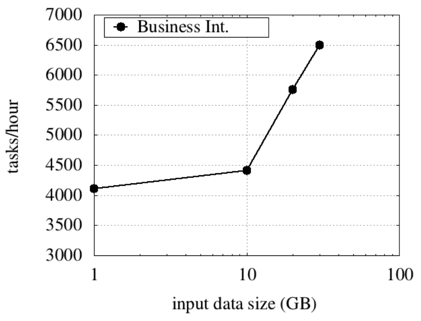
Benchmarking is an important issue for evaluating distributed systems, and extensive work has been conducted in this area. Various research and industry standard performance benchmarking solutions exist. TPC-C [32], TPC-W [33], TPC-H [27], and SPEC OMP [34] are domain-specific benchmarks: the first evaluates on-line transaction processing (OLTP) systems, the second applies to e-commerce web sites, the third evaluates decision support systems, and the fourth applies to parallel scientific computing applications that utilize OpenMP. SPECweb is another performance benchmark that evaluates web applications



(a) Request throughput



(b) Data read/write throughput



(c) MapReduce task throughput

Figure 7. Performance under different data scales

from different domains, such as banking, e-commerce and support [35].

Even though these benchmarks are useful in analyzing distributed systems in general, they are not applicable to MapReduce systems. The need of MapReduce benchmarking is motivated by the many recent works that have been devoted to the study and improvement of MapReduce performance and scalability. These include task scheduling policies in MapReduce [5], [6], [7], cost-based optimization techniques [8], replication and partitioning policies [10], [11]. All these works use microbenchmarks such as MapReduce *sort*, *grep* and *word count* programs introduced in [1]. MRBench also provides microbenchmarks in the form of a MapReduce implementation of TPC-H queries [36]. However, these microbenchmarks are not representative of full applications with complex workloads, and they do not provide automated statistics for performance.

More recent work proposed HiBench [37], a benchmark which evaluates Hadoop in terms of system resource utilization (e.g. CPU, memory), and Hadoop job and task performance. Although low-level resource monitoring may be useful in targeting specific system features, HiBench does not avoid the pitfalls of microbenchmarks, missing multi-user workloads for batch or interactive systems.

IX. Conclusion

The paper presents a comprehensive performance benchmak suite for MapReduce systems. MRBS benchmark suite adequately handles: (i) *multi-criteria analysis* with a characterization of different aspects of the performance of MapReduce systems, (ii) *diversity* in a wide range of application domains under various workloads and dataloads, and (iii) *usability* with a configurable MRBS framework portable across different cloud infrastructures and MapReduce frameworks. Evaluation of MRBS on Hadoop clusters running on Amazon EC2 successfully demonstrates these properties.

MRBS currently covers five application domains, is based on 32 MapReduce programs, and provides 26 sets of configurations and input data. Future work includes the exploration of other application domains and MapReduce programs. We hope that such a benchmark suite will help researchers and practitioners to better analyze and evaluate MapReduce systems.

X. Acknowledgments

This work was partly supported by the Agence Nationale de la Recherche, the French National Agency for Research, under the MyCloud ANR project (http://mycloud.inrialpes.fr/). Part of the experiments were conducted on the Grid'5000 experimental testbed, developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Uni- versities as well as other funding bodies (http://www.grid5000.fr).

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in The 6th Symposium on Operating System Design and Implementation (OSDI 2004), 2004.

[2] S. Chen and S. W. Schlosser, "Map-Reduce Meets Wider Varieties of Applications," Intel, Tech. Rep. IRP-TR-08-05, 2008.

[3] M. C. Schatz, "CloudBurst: Highly Sensitive Read Mapping with MapReduce," Bioinformatics (Oxford, England), vol. 25, no. 11, June 2009.

[4] "Greenplum," http://www.greenplum.com/.

[5] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in 22nd ACM Symposium on Operating Systems Principles 2009 (SOSP 2009), Big Sky, Montana, October 2009.

[6] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," in EuroSys 2010 Conference, Paris, France, April 2010.

[7] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, “Improving MapReduce Performance in Heterogeneous Environments,” in 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008), 2008.

[8] H. Herodotou and S. Babu, “Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs,” in 37th International Conference on Very Large Data Bases (VLDB 2011), 2011.

[9] A. Verma, L. Cherkasova, and R. H. Campbell, “Resource Provisioning Framework for MapReduce Jobs with Performance Goals,” in 12th ACM/IFIP/USENIX International Middleware Conference (Middleware’2011), 2011.

[10] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, “Scarlett: Coping with Skewed Content Popularity in MapReduce Clusters,” in EuroSys 2011 Conference, Salzburg, Austria, April 2011.

[11] M. Eltabakh, Y. Tian, F. Ozcan, R. Gemulla, A. Krettek, and J. McPherson, “CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop,” in 37th International Conference on Very Large Data Bases (VLDB 2011), Seattle, Washington, September 2011.

[12] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Rasin, and A. Silberschatz, “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads,” in 35th International Conference on Very Large Data Bases (VLDB 2009), Lyon, France, August 2009.

[13] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, “Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing),” in 36th International Conference on Very Large Data Bases (VLDB 2010), Singapore, September 2010.

[14] A. Floratou, J. Patel, E. Shekita, and S. Tata, “Column-Oriented Storage Techniques for MapReduce,” in 37th International Conference on Very Large Data Bases (VLDB 2011), Seattle, Washington, September 2011.

[15] M.-Y. Lu and W. Zwaenepoel, “HadoopToSQL,” in EuroSys 2010 Conference, Paris, France, April 2010.

[16] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears, “MapReduce Online,” in The 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’10), 2010.

[17] H. Liu and D. Orban, “Cloud mapreduce: A mapreduce implementation on top of a cloud operating system,” in The 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID ’11), Washington, DC, 2011.

[18] “Apache Hadoop,” http://hadoop.apache.org/.

[19] “Amazon Elastic MapReduce,” http://aws.amazon.com/elasticmapreduce/.

[20] “Google App Engine,” http://code.google.com/intl/en/appengine/.

[21] “Open Cirrus: The HP/Intel/Yahoo! Open Cloud Computing Research Testbed,” https://opencirrus.org/.

[22] “HDFS: Hadoop Distributed File System,” http://hadoop.apache.org/hdfs/.

[23] “Amazon Elastic Compute Cloud (Amazon EC2),” http://aws.amazon.com/ec2/.

[24] “OpenStack,” http://www.cloud.com.

[25] “MovieLens web site,” http://movielens.umn.edu/.

[26] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, Recommender Systems: An Introduction. Cambridge University Press, 2010.

[27] “TPC Benchmark H - Standard Specification,” http://www.tpc.org/tpch/.

[28] “Apache Hive,” http://hive.apache.org/.

[29] “Genomic research centre,” http://www.sanger.ac.uk/.

[30] “Apache Mahout machine learning library,” http://mahout.apache.org/.

[31] “20 Newsgroups,” http://people.csail.mit.edu/jrennie/20Newsgroups/.

[32] “TPC-C: an on-line transaction processing benchmark,” http://www.tpc.org/tpcc/.

[33] “TPC-W: a transactional web e-Commerce benchmark,” http://www.tpc.org/tpcw/.

[34] Standard Performance Evaluation, “SPEC OpenMP Benchmark Suite,” http://www.spec.org/omp/.

[35] “SPECweb2009,” http://www.spec.org/web2009/.

[36] K. Kim, K. Jeon, H. Han, S.-g. Kim, H. Jung, and H. Y. Yeom, “MRBench: A Benchmark for MapReduce Framework,” in 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS ’08), 2008.

[37] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, “The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis,” in 22nd International Conference on Data Engineering Workshops (ICDE 2010), Los Alamitos, CA, 2010.