Modeling the Performance of MapReduce Applications for the Cloud

Iván Carrera and Cláudio Geyer

***Abstract*—In the last years, Cloud Computing has become a key technology that made possible to run applications without needing to deploy a physical infrastructure. The challenge with deploying distributed applications in Cloud Computing environments is that the virtual machine infrastructure should be planned in a time and cost-effective way.**

This work is a summary of a previous work presented by the authors as a Master's thesis, with the goal of showing that the execution time of a distributed MapReduce application, running in a Cloud computing environment, can be predicted using a mathematical model based on theoretical specifications. This prediction is made to help the users of the Cloud Computing environment to plan their deployments, i.e., quantify the number of virtual machines and its characteristics. After measuring the application execution time and varying parameters stated in the mathematical model, and after that, using a linear regression technique, the goal is achieved finding a model of the execution time which was then applied to predict the execution time of MapReduce applications. Experiments were conducted in several configurations and showed a clear relation with the theoretical model, revealing that the model is in fact able to predict the execution time of MapReduce applications. The developed model is generic, meaning that it uses theoretical abstractions for the computing capacity of the environment and the computing cost of the MapReduce application.

***Index Terms*—MapReduce, Cloud, Hadoop, FLOPs, MRBS, Performance**

***Resumen*—En los ltimos aos, Cloud Computing se ha convertido en una tecnologa clave que ha hecho posible ejecutar aplicaciones sin la necesidad de utilizar una infraestructura fsica. El desafo de implementar aplicaciones distribuidas en ambientes de Cloud Computing es que la infraestructura de mquinas virtuales debe considerar aspectos relacionados con el costo y el tiempo de utilizacin. Este trabajo es el resumen de uno anterior, presentado por los autores como tesis de maestra, con el objetivo de demostrar que el tiempo de ejecucin de una aplicacin distribuida MapReduce, ejecutndose en un ambiente de Cloud Computing, puede ser predicho utilizando un modelo matemtico basado en especificaciones tericas. Esta prediccin se realiza para ayudar a los usuarios de un ambiente de Cloud Computing a planificar sus implementaciones, es decir, cuantificar el nmero de mquinas virtuales y sus caractersticas. Despus de medir el tiempo de ejecucin de las aplicaciones y variando los parmetros establecidos por el modelo matemtico, y seguidamente usando una tcnica de regresin lineal, el objetivo se alcanza al encontrar un modelo del tiempo de ejecucin que fue posteriormente aplicado para aplicaciones MapReduce. Los experimentos fueron realizados en diferentes configuraciones y mostraron una clara relacin con el modelo terico, mostrando as que el modelo es capaz de predecir el tiempo de ejecucin de aplicaciones MapReduce. El modelo desarrollado es genrico, es decir que usa abstracciones tericas para la capacidad de cmputo del ambiente y el costo computacional de la aplicacin MapReduce.**

***Palabras clave*—MapReduce, cloud, hadoop, FLOPs, MRBS, rendimiento**

I. Introduction

Cloud Computing, as defined in [1] by the U.S. National Institute of Standards and Technology NIST, is a computing model that enables ubiquitous, on-demand network access to a shared pool of configurable computing resources. Likewise, the U.S. Department of Energy, DoE, says in [2] that a Cloud Computing model still lacks a good performance management, even though it can bring to the general user a big gain in terms of scalability, and usability of a computing infrastructure.

MapReduce is a programming model and an associated implementation for processing and generating large data-sets proposed by Google [3]. MapReduce performance depends, amongst other things, on the type of data that is going to be processed, so various types of workloads can have different characteristics.

Research presented in [4] establishes a number of parameters that have to be configured by the system administrators of a MapReduce application, in a way that is time-effective. As it is discussed in works as [5] and [6], an optimal cluster size should save money by optimizing infrastructure Cloud resources for the MapReduce user and the Cloud provider. This optimization can let users to take a better advantage of the resources they are using in the Cloud.

The research in this paper is the summary of a Master's thesis presented by the authors in [7], and motivated by the need of being able to predict the execution time of a distributed application running in a Cloud Computing environment, that would allow users to perform Capacity Planning, and so plan their virtual clusters in a way that is cost and time-effective. The main objective of this paper is to present a formula for predicting the execution time of a MapReduce application. This research's main approach is to develop a prediction model as was described in [8].

The prediction will be done with a formula that computes the execution time t of the application as a function of four variables:

1) W, the amount of workload;
2) p, the number of virtual machines the application will be run on top of, since it is a distributed application;
3) A, the type of the MapReduce application, since Map Reduce is a very versatile framework and many different

Iván Carrera. Department of Informatics and Computer Science National Polytechnic School (EPN). Isabel La Católica s/n, Quito, Ecuador. ivan.carrera@epn.edu.ec

Cláudio Geyer. Institute of Informatics (INF). Federal University of Rio Grande do Sul (UFRGS). Av. Bento Gonçalves, 9500, Porto Alegre, Brazil. geyer@inf.ufrgs.br

applications can be programmed with it. *A* will be expressed as a relation between the number of operations per GB the application has to perform in order to process the *W* amount of data in Map and Reduce phases; and,

4) *T*, the capacity of the *p* computers that compose the cluster on top of which the application will be run. *T* will be expressed in terms of the number of operations per second that every computer in the cluster is capable of perform.

Thus,

$$t = f(W, p, A, T) \quad (1)$$

This work is intended to help users to know *'a priori'* how much time it will take for the application to run, so they can plan their virtual machine clusters, and find a best-suited configuration for their MapReduce application.

The experimental results of this work are a valuable contribution, showing that it is possible to model the execution time of a MapReduce application, and how the performance of a MapReduce application in terms of execution time changes when running in different environment configurations.

The remainder of this paper is organized as follows: section II talks about the related research works about modeling the performance of MapReduce applications, section III addresses the theoretical mathematical model used for this work, section IV explains the performed experiments, how they were designed and executed, section V discusses the experimental results and how they prove our theoretical approach, and finally, section VI talks about the Conclusions of this work and suggests the future work for further research.

II. Related Work

Extensive and comprehensive models for each phase of MapReduce are presented in [9]. In said work, the MapReduce algorithm has two sets of tasks: Map, run first, and Reduce, run last. The Map task execution is divided into five phases: Read, Map, Collect, Spill and Merge; and in the same way, the Reduce task execution is divided into four phases: Shuffle, Merge, Reduce and Write. In [9], the execution time is calculated as a function of three parameters: *d*, data properties, *r*, cluster resource properties, and *c*, configuration parameter settings. It also shows a set of formulas that are useful to determine the execution time of a MapReduce job, introducing the concept of 'cost' for the stages, understood as what determines the amount of performed operations on the CPU or the hard drive to process data or write/read data from/to the hard drive. The Technical Report does not go further and does not assess the models with experimentation. The models are exclusively theoretical.

Another interesting work in the topic of MapReduce and Performance Evaluation is [10]. In said work, authors present three cost functions that show a relationship between some characteristics of the MapReduce application and the time that takes for the application to execute. These three cost functions for MapReduce differ from each other by taking into account more or less complexity of the MapReduce application; for example, the simplest approach assumes that the amount of input data is fixed, so it is not necessary to take into account in the cost function. Authors assess their performance model taking into account specific parameters of MapReduce applications like the number Map and Reduce slots, the number Map and Reduce rounds, the complexity of Map and Reduce phases and the cost of scheduling all said processes.

Other work showing performance models of MapReduce is [11], where MapReduce is modeled in three separated phases: Map, Shuffle and Reduce. Also, said work takes into account three parameters: *memory*, understood as the capacity of a cluster node to save information in its hard disk, *machine*, understood as the total number of nodes composing the cluster and *time*, as the running time available for the execution of the MapReduce application. Authors describe also the execution of a MapReduce job subject to a probability function of correctness, meaning that the job will be executed with no error in only some cases, defined by the probability function.

Authors in [12] determine 5 design factors of a cluster utilized to run MapReduce application that affect the performance, namely:

1) **I/O mode.** Which is the way the MapReduce application gets its input data: *direct* mode, when reading directly from the hard drive or *streaming* mode, when streaming by a communication scheme as TCP/IP or JDBC,
2) **Indexing.** Even if MapReduce is typically used for unsorted data, when using sorted files, or database indexed tables, performance seems to improve,
3) **Data parsing.** If there is any decoding procedures inside the Map phase that can be fixed or variable along the execution,
4) **Grouping schemes** of input data, and
5) **Block-level scheduling.** Showing that the scheduler performs faster when using larger blocks.

Authors also investigated alternative implementation strategies for each factor, and how they affect the general performance of MapReduce applications. They have evaluated the performance of MapReduce with representative combinations of these five factors using a benchmark consisting of seven tasks. Amongst their findings is that MapReduce achieves elastic scalability through block-level scheduling.

III. Proposed Model

Hadoop [13] is a Java implementation of MapReduce proposed by the Apache Foundation. It is the most used MapReduce implementation [14]. Hadoop's API (Application Programming Interface) allows the user to program the Map and the Reduce functions.

Hadoop was originally thought to be run as a distributed application on top of a cluster of homogeneous machines, sharing a distributed filesystem. Hadoop uses a distributed filesystem called the Hadoop Distributed FileSystem HDFS, which is similarly an open implementation of the Google FileSystem GFS described in [3]. HDFS performs similarly to Google File System [15]. HDFS is based on NFS, working as a shared partition, accessible from all machines, in every hard disk of the nodes composing the MapReduce cluster.

The Hadoop execution algorithm is based on the MapReduce proposed by Google. According to the Hadoop execution

algorithm [13], represented in figure 1 [14], it has four defined phases:

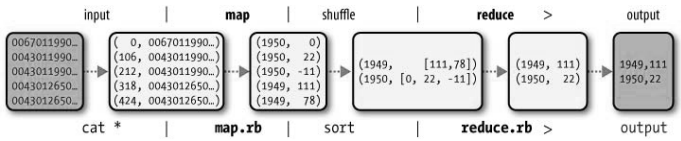


Fig. 1. MapReduce execution flow

1. Data distribution. The first process is to copy the data to the HDFS. This is done by the **master** node. The time length of this step depends on three factors: (1) the amount of data that has to be copied to the cluster nodes, (2) the number of nodes, and (3) the speed of the network that connects said nodes. Normally, all nodes would receive approximately the same amount of data workload, because the approach used is write-once read many.

2. Map phase. The second process is mapping the data, taking the input data and generating an output composed of *<key, value>* pairs according to the programming of the Map phase. Each node processes the data that it has locally stored. Since the Map phase is run by all the nodes in parallel, the time length of this phase would depend on the amount of workload a single node has to process, the speed of said node to process said individual workload, and the number of operations that involve the processing of the individual workload.

3. Shuffle. Once the Map phase has reached a 5% of its progress, map outputs are sorted to ease the processing in the next phase. Output data from the Map phase is transmitted over the network to the data nodes according to its content. The time length of this phase depends on the amount of data is transmitted and the speed of the network. This phase runs in parallel with the Map phase, and for matters of timing, we can only takes into account the shuffling of the last map outputs.

4. Reduce phase. Finally, the reduce phase takes the output of the shuffle phase and process it according to its programming. The time length of this phase depends on the amount of data that is going to be processed, the number of operations that would be required to process the Reduce inputs, again, the speed of the node.

Following, we propose propose an equation that can quantify the execution time of a Hadoop application, based on the aforementioned Hadoop algorithm.

As said in section I, there are 4 parameters for the evaluation of the MapReduce application, namely:

1) *W*, the amount of workload the application is intended to process, expressed in units of storage;
2) *p*, the number of computers the application will be run on top of, since it is a distributed application, it is supposed to be run on top of a cluster of computers, expressed as an integer;
3) *A*, the type of the MapReduce application, since MapReduce is a very versatile framework and many different applications can be programmed with it. *A* will be expressed as a the number of operations per unit of storage performed by the application in Map and Reduce phases; and,
4) *T*, the processing capacity of the p computers that compose the cluster on top of which the application will be run. *T* will be expressed in terms of the number of operations per second that every machine in the cluster is capable of perform.

In order to build a mathematical model of the MapReduce execution time, we have to make some considerations:

- The *Data Distribution* phase is not considered inside the execution time of the Hadoop program, so its duration can be disregarded;
- The *Shuffle* phase starts when the Map phase has reached a 5% of its completion, and runs simultaneously with the Map phase;
- For matters of simplifying the model, the *Shuffle* phase time will be considered only as the time when the *Map* phase has already finished;
- The *Reduce* phase starts only when the *Shuffle* phase has been fully completed;
- The *Map* and *Reduce* phases run in *slots*, that means that each node can run more than one Map or Reduce task in parallel; usually, the number of slots is related with the number of cores a node has.

Thus, the execution time of a Hadoop program can be modeled as the sum of the times of all phases:

$$t_{total} = t_{MAP} + t_{SHUFFLE} + t_{REDUCE} \tag{2}$$

The time of the Map phase t_{MAP} can be computed as the product of the time of a single Map task times the number of Map tasks per node:

$$t_{MAP} = t_{U_{MAP}} \cdot \frac{n_{MAP}}{p} \tag{3}$$

The time of a single Map task $t_{U_{MAP}}$ is computed as the product of the amount of workload for a single Map task W_{MAP} (known as a *chunk size*) times the cost of a single Map task $C_{U_{MAP}}$ (expressed as a relation of the number of floating point operations FLOPs required by the Map task and the chunk size), and divided by the capacity *T* of a single node (the number of floating point operations FLOPs per second):

$$t_{U_{MAP}} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \tag{4}$$

The number of Map tasks for a job is computed as the division between the total Workload and the *chunk size*:

$$n_{MAP} = \left\lceil \frac{W}{chunksize} \right\rceil \tag{5}$$

The time of the Reduce phase t_{REDUCE} can be computed as the product of the time of a single Reduce task times the number of Reduce tasks per node:

$$t_{REDUCE} = t_{U_{REDUCE}} \cdot \frac{n_{REDUCE}}{p} \tag{6}$$

The time of a single Reduce task $t_{U_{REDUCE}}$ is computed as the product of the amount of input workload for a single Reduce task $W_{U_{REDUCE}}$ times the cost of a single Reduce task $C_{U_{REDUCE}}$, and divided by the capacity *T* of a single node (the number of floating point operations FLOPs per second):

$$t_{U_{REDUCE}} = \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T} \tag{7}$$

Replacing respectively equations 4 and 7 into equations 3 and 6 we have:

$$t_{MAP} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \cdot \frac{n_{MAP}}{p} \quad (8)$$

$$t_{REDUCE} = \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T} \cdot \frac{n_{REDUCE}}{p} \quad (9)$$

Also, the time for the Shuffle phase $t_{SHUFFLE}$ can be computed as the product of the size of the output data of a single Map task times the number of remaining Shuffle tasks divided by the bandwith of the network:

$$t_{SHUFFLE} = \frac{W^{OUT}_{U_{MAP}} \cdot (n_{MAP} mod p)}{B} \quad (10)$$

The size of the output data of a single Map task $W^{OUT}_{U_{MAP}}$ can be computed as the division between the total amount of output data of the Map phase divided by the number of Map tasks.

$$W^{OUT}_{U_{MAP}} = \frac{W^{OUT}_{MAP}}{n_{MAP}} \quad (11)$$

So, replacing equations 8, 9 and 10 in equation 2, and simplifying W^{IN}_{REDUCE}, we have:

$$t_{total} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \cdot \frac{n_{MAP}}{p} + \frac{W^{OUT}_{MAP} \cdot (n_{MAP} mod p)}{n_{MAP} \cdot B} + \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T \cdot p}$$

Equation III will serve as the performance model for MapReduce applications as a function of the parameters of the distributed system.

IV. Experiments

A. Scenarios

In order to verify the model of equation III, it is required that experiments are performed in several different environments, comprising private and public infrastructure, physical and virtual cloud infrastructure. The environments where the tests were run in, are described following:

1) **Cluster `gradep`**
   The `gradep` is a computer cluster on premises of Institute of Informatics *INF*, in the Federal University of Rio Grande do Sul *UFRGS*, composed of 18 nodes in total. Each node has an Intel Pentium 4 2.79 GHz CPU with 2 GB in RAM, and a Gigabit Ethernet connection. Each time the applications were run, the worker nodes were randomly chosen amongst these 18 machines to avoid errors produced by running all times in the same cluster nodes.
2) **Amazon EMR**
   Amazon Elastic MapReduce EMR [16] is a web service from Amazon Cloud services that uses Hadoop to process data across a cluster of Amazon EC2 instances [17].
   For the purposes of this research work, the `m1.small` instance type was used. EC2 instance types are specified in [17]. The processing power of Amazon EC2 instances is expressed in ECU, which is a unit with the equivalent CPU power of a 1.0-1.2 GHz 2007 Opteron or Xeon processor as specified to Amazon EC2 documentation. The `m1.small` instance type, for example, has the processing power of 1 ECU.
   According to [18], the theoretical peak performance can be computed for different instances from the ECU definition: a 1.1 GHz 2007 Opteron can perform 4 flops per cycle at full pipeline, which means at peak performance one ECU equals 4.4 gigaflops per second.
3) **Windows Azure HDInsight**
   Windows Azure HDInsight [19] is a Hadoop-based service from Microsoft Azure for running an Apache Hadoop solution in a Cloud Computing environment. HDInsight uses the General Purpose Instances from Microsoft Azure.
   In this work we used the Large (A3) compute instance type. According to the Windows Azure documentation, the A3 compute instance type has a 4 cores 1.6 GHz processor, and 7GB in RAM.

B. Software description

Varying the applications in the tests help the model to identify when it cannot be useful, or if it only serves for a certain type of applications. The applications used for the tests are described following:

- In the `gradep` cluster, two applications, named `sort` and `wordcount`, were used. These applications form part of the `text-processing` benchmark from the MapReduce Benchmark Suite *MRBS* described in [20].
  As it is explained in [21], the `sort` application takes an input of text registers, and sorts them depending on its contents. And the `wordcount` application, as explained in [3], takes a text input and counts the occurrences of each word, resulting in a sorted list of words indicating how many times they appeared in the input text.
  The `wordcount` application is *CPU bound*, meaning that the execution time is mainly limited by the processing power of the CPU, and the `sort` application is *CPU bound* and *IO Bound* were used, meaning that the execution time is mainly limited by the CPU and the IO system.
- And finally, in the Cloud environments of *Amazon* and *Azure* it was used a log processing application, which basically is a text processing application similar to the ones used in the environments explained above.
  In 2013, a private company located in Brazil, asked the GPPD/INF research group for help to develop and test a MapReduce application to process large amounts of logs and to be run in a Cloud environment. The project supported the research presented in this dissertation by providing a use case for what it is intended to do, to predict the execution time of a MapReduce application running in a Cloud environment. The log processing application takes the log records and rearranges the contents of each text line, leaving the output slightly smaller than the output. The logs contain the same information but in a different order. More information about the application and the project can be found in [22].

C. Specification of Experiments

1) Evaluation Technique: In the present work we used two evaluation techniques, one for develop the performance model in equation III, and another one to assess said performance model. *Analytical Modeling* was first used as an evaluation technique, where, based on theoretical models of MapReduce, a formula to model its performance was developed. The second evaluation technique used in this work is *Experimental Measurement*. In this technique, the analytical/theoretical hypothesis is tested with experimentation.

2) Performance Metrics: As it has been discussed earlier in this work, and modeled in equation III, the performance metric used in the experiments is the execution time of the applications, considered from the beginning of the first Map operation, until the end of the last Reduce operation.

3) Workload: As said in [23], when the workload used in the experiments is a typical application, and it is applied to test a set of environments, it is called a benchmark.

4) Parameter Values: The values for the 4 parameters described in section I depend on the scenarios described in section IV-A.

Table I shows the used values in the different scenarios.

TABLE I
PARAMETER VALUES

Scenario	`gradep` cluster	AWS	Azure
W	{1, 5, 10, 20, 25} [GB]	{1, 5, 10, 20, 25} [GB]	{1, 5, 10, 20, 25} [GB]
p	{4, 8, 12, 16} [nodes]	{4, 6, 8} [nodes]	{4, 6, 8} [nodes]
A	text processing	log processing	log processing
T	`gradep`	m1.small	A3

In table I, *W*, the amount of workload is expressed in GB; *p*, the number of nodes is an integer; *A*, the type of the Map Reduce application is expressed as an application type, while in the formula this categorical value has to be changed for a numeric one in $[flops/GB]$; and *T*, the node capacity is expressed as the name of the cluster the node belongs to, or the instance type if it belongs to a Cloud environment, while in the formula this categorical value has to be changed for a numeric one in $[flops]$.

V. RESULTS

A. Private Cluster `gradep`

As shown in table I, in the `gradep` cluster, two applications were run to obtain and test the performance model. These applications were taken from the text-processing benchmark of MRBS [20]. Values for *W* and *p* in table I were arbitrarily established in function of the available nodes in `gradep` cluster. Each combination was run 20 times, for a total of 800 executions.

After the executions for the `sort` application, we noticed that the Map phase output W_{MAP}^{OUT} is proportional to the Map phase input W_{MAP}. This makes sense because the amount of Map output data would depend on the amount of Map input data, and the following relation can be established:

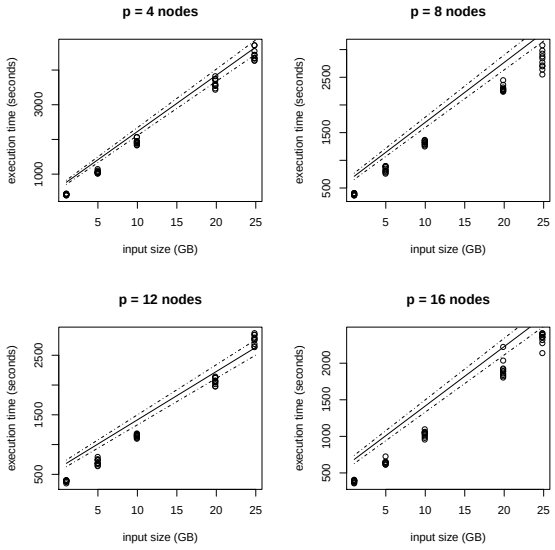


Fig. 2. Execution time vs. Workload for the `gradep` cluster

$$W_{MAP}^{OUT} \propto W_{MAP} \tag{12}$$

$$W_{MAP} = kW_{MAP}^{OUT} \tag{13}$$

And when replacing equation 13 into equation III, it simplifies to:

$$\mathrm{t}_{total} = \frac{W_{MAP}}{pT} \cdot (C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}) + \frac{W_{MAP}^{OUT} \cdot (n_{MAP} mod p)}{n_{MAP} \cdot B}$$

Without making these simplifications in equation III, we could not be able to calculate the values for *A*. Equation V-A shows the linear relation between the execution time t and the workload W_{MAP}.

Also, since it was required to quantify the computational power of the cluster nodes, that can be done using a benchmark. However, the tests were already performing a benchmark over the cluster, so, with the experimental data, it was possible compute the relation between the cost for Map and Reduce phases $C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}$ and the computational power of the nodes T using the linear regression tool of the software *R* [24]. The value of the relation between the cost for Map and Reduce phases and the computational power was computed to:

$$\frac{C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}}{T} = 6.05e^{-7} \left[\frac{s}{Byte}\right] \tag{14}$$

With an standard error of $SE = 1.15e^{-8}$ and a coefficient of determination of $R^2 = 0.88$. A value of $R^2 = 0.88$ gives us the confidence that the model is pretty close to an optimal fit, but considering this model being general and that it has to serve different values, it can be considered as a very good model.

In figure 2 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines

TABLE II
RESULTS FOR MODEL ASSESSING ON THE GRADEP CLUSTER

Values	Predicted Value	Exp. Value	Error
W=1, p=6	401.6	430.7	7.24%
W=1, p=10	373.4	374.8	0.38%
W=1, p=14	339.7	387.3	14.0%
W=5, p=6	899.3	962.1	6.98%
W=5, p=10	752.9	734.7	2.42%
W=5, p=14	687.9	653.1	5.05%
W=10, p=6	1521.4	1442.0	5.22%
W=10, p=10	1227.3	1241.0	1.12%
W=10, p=14	1123.0	1250.3	11.33%
W=20, p=6	2765.6	2526.2	8.66%
W=20, p=10	2176.0	2454.4	12.8%
W=20, p=14	1993.3	2021.5	1.41%
W=25, p=6	3387.7	3392.4	0.14%
W=25, p=10	2560.3	2557.0	3.52%
W=25, p=14	2428.5	2296.9	5.42%

showing the 95% confidence interval of the model. Figure 2 shows that the experimental results follow a clear trend that is followed by the model.

After modeling the execution time, giving values to the formula of equation V-A, more experiments were run to assess the model, with the results detailed in table II.

Results in table II show that the model from equation V-A, using the values in equation 14, was actually able to predict the execution time of the MapReduce applications with a reasonable error. This error is small considering that when expressed in absolute value it reduces to a few minutes of execution. We have to keep in mind that these models are meant to be used in Cloud infrastructures, where a few minutes of difference do not mean excessive changes in the cost of using the platforms.

B. Amazon Elastic MapReduce

The experiments using the cloud infrastructure of Amazon Elastic MapReduce followed the values shown in table I. Each combination was run 10 times, for a total of 150 executions.

The main restrictions for only running 10 times was the budget and the fact that the account for running the tests in Amazon Web Services belonged to the enterprise that asked the GPPD/INF research group for assistance as it was explained in subsection IV-B and in [22].

The log processing application, as explained in section IV-B is an example of a sort application, meaning that the relation explained in equation 12 serves in this case as well. So, the model to follow with this environment is the one explained in equation V-A.

The value of the cost for Map and Reduce phases was computed using the software *R* [24] to:

$$C_{U_{MAP}} + k \cdot C_{U_{REDUCE}} = 2.16e^6 \left[\frac{flop}{Byte}\right] \qquad (15)$$

With an standard error of $SE = 6.46e^4$ and a coefficient of determination of $R^2 = 0.88$.

Following, in figure 3 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines showing the 95% confidence interval of the model.

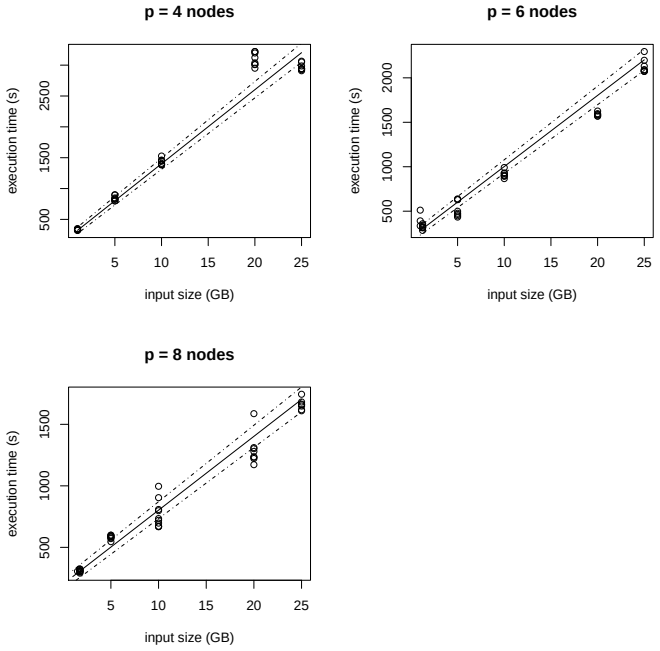


Fig. 3. Execution time vs. Workload for the Amazon EMR Environment

After modeling the execution time, giving values to the formula of equation V-A, more experiments were performed in order to assess the model, with the results described in table III:

TABLE III
RESULTS FOR MODEL ASSESSING OF THE AMAZON MEASURES

Values	Predicted Value	Exp. Value	Error
W=1, p=5	531.01	458.22	13.71%
W=1, p=7	488.23	361.53	25.95%
W=5, p=5	1061.4	870.24	18.01%
W=5, p=7	860.64	780.91	9.26%
W=10, p=5	1724.38	1556.93	9.71%
W=10, p=7	1326.15	1186.78	10.51%
W=20, p=5	3050.35	2826.88	7.33%
W=20, p=7	2257.17	2928.47	29.74%
W=25, p=5	3713.33	3556.38	4.23%
W=25, p=7	2722.68	2414.57	11.32%

Table III shows the percentage error from comparing the execution times obtained. Errors show a gap between experimental and theoretical values, however, considering that in a Cloud environment the execution times are charged by the hour or fraction, this values should be considered important only when reaching values close to one hour.

C. Windows Azure HDInsight

Experiments using the Windows Azure HDInsight cloud infrastructure used values from the table I. Each combination was run 10 times, for a total of 150 executions.

Similarly to the experiments in Amazon Elastic MapReduce, the main restrictions for only running 10 times each combination of parameter values were the budget and the fact that we were not the owners of the account for running the tests.

As in the previous case, the log processing application being an example of a sort application, follows the model explained in equation V-A. And, since there was no data to assign a value to T for the A3 Azure node type, the relation between the cost for Map and Reduce phases $C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}$ and the computational power T was computed to:

$$\frac{C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}}{T} = 2.16e^{6} \left[\frac{s}{Byte}\right] \qquad (16)$$

With an standard error of $SE = 1.06e^{-5}$ and a coefficient of determination of $R^2 = 0.86$.

In figure 4 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines showing the 95% confidence interval of the model.

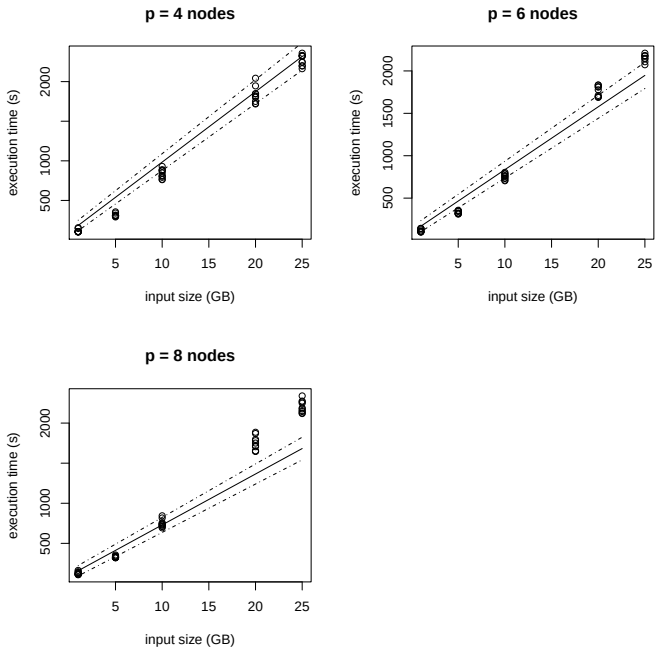


Fig. 4. Execution time vs. Workload for the Azure HDInsight Environment

After the execution time model was obtained, giving values to the formula of equation V-A, more experiments were run to assess the model and to verify that it was able to predict the performance of a MapReduce application running in the Azure HDInsight cloud environment. Each combination of W and p was run 10 times as well, in order to obtain the average and error values shown in table IV.

The values in table IV show the percentage error from comparing the execution times obtained. As in the previous cases, since experiments deal with commercial Cloud environments, where execution time is charged by the hour or fraction, this values should be considered big when reaching values close to one hour.

TABLE IV
RESULTS FOR MODEL ASSESSING THE MEASURES ON THE AZURE HDINSIGHT ENVIRONMENT

Values	Predicted Value	Exp. Value	Error
W=1, p=6	401.6	430.7	7.24%
W=1, p=10	373.4	374.8	0.38%
W=1, p=14	339.7	387.3	14.0%
W=5, p=6	899.3	962.1	6.98%
W=5, p=10	752.9	734.7	2.42%
W=5, p=14	687.9	653.1	5.05%
W=10, p=6	1521.4	1442.0	5.22%
W=10, p=10	1227.3	1241.0	1.12%
W=10, p=14	1123.0	1250.3	11.33%
W=20, p=6	2765.6	2526.2	8.66%
W=20, p=10	2176.0	2454.4	12.8%
W=20, p=14	1993.3	2021.5	1.41%
W=25, p=6	3387.7	3392.4	0.14%
W=25, p=10	2560.3	2557.0	3.52%
W=25, p=14	2428.5	2296.9	5.42%

As we can see from the results of tables III and IV, it is possible to predict the performance of a MapReduce application running in a Cloud Computing environment, thus achieving the main goal of this work.

VI. CONCLUSIONS AND FURTHER WORK

This research work was successful when predicting the execution time of MapReduce applications in the Cloud. A mathematical model with the form: $t = f(W, p, A, T)$ was developed and specified in some cases.

The mathematical model is based on the Hadoop Algorithm model described in section III. The formulae expressed in the same section was able to compute the execution time of MapReduce applications, for the cases described in section IV.

Results of section V show that there is an error rate when assessing the predictions made by the formula. It means that some further work, focusing in tuning the formula, should be done. Also, the shown simplifications of the formula allow us to see more easily the factors that have a direct impact on the execution time of the MapReduce applications.

The formula expressed in equation III is a suitable model for the execution time of MapReduce applications. Amongst its strengths we can say that it is a general model, since it does not depend on a type of application or hardware, it models the execution time with general parameters like the computing power of nodes, expressed as a number of operations per second performed by their processors, and the computing cost of Map and Reduce phases, expressed as the number of operations that the nodes have to process per unit of storage. This model should be able to compute the execution time of a large set of MapReduce applications.

In this work we have proven the idea explained in [8] where we said that following a performance evaluation methodology, we should be able to predict the execution time of a distributed application in a Cloud environment and, conversely, given a time constraint we could be able to know what virtual infrastructure can be enough to execute the distributed application.

Being able to predict the execution time of distributed applications in the Cloud can be useful when preparing a

budget. A MapReduce user can calculate the time and know how much time it will take to run the application.

As it was stated in the motivation of this work, further work is encouraged to test the approach described in this work with different distributed applications, and different scenarios. Also, further research can be done in order to improve the services of Cloud providers, namely:

From the point of view of a Cloud provider, being able to tell the time that a distributed application will take to run *'a-priori'* is useful in business models like the Spot Instances of Amazon [17]. In said business model, providers offer virtual machines with lower prices, based on the fact that they were reserved for other users for a longer time than the one they were actually busy. So, the provider can re-lease the virtual machines in lower prices. If a provider is able to know when a virtual machine will be freed, it can predict when it can be used as a Spot Instance.

With the models exposed in section V, a Cloud provider can use the approach explained in this work to program a feature of its Cloud platform where a Cloud user can know how much time a given MapReduce application will take to run, letting him program his budget.

Some other improvements in Cloud services can be performed, and hopefully this work can serve as a basis of a subject that could represent an important advance in Cloud services.

ACKNOWLEDGMENTS

This work was made with the support of the *Programa Estudantes-Convênio de Pós-Graduação (PEC-PG)*, of CAPES/CNPq - Brazil.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST special publication*, vol. 800, p. 145, 2011.

[2] K. Yelick, S. Coghlan, B. Draney, R. S. Canon *et al.*, "The magellan report on cloud computing for science," *US Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR) December*, 2011.

[3] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] S. Babu, "Towards automatic optimization of mapreduce programs," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 137–142.

[5] H. Herodotou, F. Dong, and S. Babu, "No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 18.

[6] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computational models and the heterogeneity challenge," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 77–86, 2012.

[7] I. Carrera Izurieta and C. Geyer, "Performance modeling of mapreduce applications for the cloud," Master's thesis, Universidade Federal do Rio Grande do Sul, 2014. [Online]. Available: "http://hdl.handle.net/10183/99055"

[8] I. Carrera and C. Geyer, "Impressionism in cloud computing. a position paper on capacity planning in cloud computing environments," in *Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS)*. INSTICC, 2013, pp. 333–338.

[9] H. Herodotou, "Hadoop performance models. technical report cs-2011-05," *Duke Computer Science*, 2011. [Online]. Available: "http://www.cs.duke.edu/starfish/files/hadoop-models.pdf"

[10] F. Tian and K. Chen, "Towards optimal resource provisioning for running mapreduce programs in public clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 155–162.

[11] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for mapreduce," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010, pp. 938–948.

[12] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of mapreduce: An in-depth study," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 472–483, 2010.

[13] Hadoop, 2013, apache Hadoop https://www.grid5000.fr/ accessed on 12/28/2013.

[14] T. White, *Hadoop: the definitive guide*. O'Reilly, 2012.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[16] EMR, 2013, amazon Web Services - EMR Elastic MapReduce http://aws.amazon.com/elasticmapreduce accessed on 07/23/2013.

[17] EC2, 2013, amazon Web Services - EC2 Elastic Compute Cloud http://aws.amazon.com/ec2 accessed on 07/23/2013.

[18] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 931–945, 2011.

[19] HDInsight, 2013, windows Azure HDInsight http://azure.microsoft.com/en-us/documentation/services/hdinsight/ accessed on 12/02/2014.

[20] A. Sangroya, D. Serrano, and S. Bouchenak, "Benchmarking dependability of mapreduce systems," in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*. IEEE, 2012, pp. 21–30.

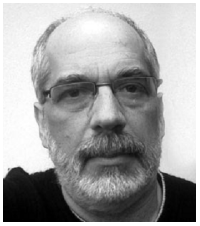
[21] O. OMalley, "Terabyte sort on apache hadoop," *Yahoo, available online at: http://sortbenchmark. org/Yahoo-Hadoop. pdf*, pp. 1–3, 2008.

[22] I. Carrera, F. Scariot, P. Turin, and C. Geyer, "An example for performance prediction for map reduce applications in cloud environments," in *Escola Regional de Redes de Computadores ERRC - RS Rio Grande do Sul*, 2013.

[23] R. Jain, *The art of computer systems performance analysis*. John Wiley & Sons Chichester, 1991, vol. 182.

[24] R, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0 http://www.R-project.org/.

Iván Marcelo Carrera Izurieta is an assistant professor at the Department of Computer Science and Informatics of the National Polytechnic School. His research interest include parallel and distributed computing, performance evaluation and innovation. He received his MSc in Computer Science from the Informatics Institute of the Federal University of Rio Grande do Sul. He's a member of the Brazilian Computing Society. Contact him at Escuela Politécnica Nacional, Quito, Ecuador; ivan.carrera@epn.edu.ec.



Cláudio Fernando Resin Geyer is an associate professor at the Informatics Institute of the Federal University of Rio Grande do Sul. His research interests include ubiquitous computing, parallel and distributed computing, grid computing, and distributed objects. He received his PhD in informatics from the Joseph Fourier University. Hes a member of the ACM and the Brazilian Computer Society. Contact him at Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil; geyer@inf.ufrgs.br.