A Use Case for Performance Prediction with Map Reduce Applications

Pierre Turin
Ensimag - Grenoble INP
681 rue de la Passerelle - BP 72
F-38402 St Martin d'Hères Cedex, France
Pierre.Turin@ensimag.grenoble-inp.fr

Iván Carrera
Institute of Informatics INF
Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre - RS - Brazil
ivan.carrera@inf.ufrgs.br

Abstract

Map Reduce is a programming framework for applications that process and generate large data sets, intended to be run on multi-computer environments. The goal of this paper is to expose a use case for a methodology for finding a model of the execution time of a Map Reduce application. After measuring the time taken by the application and varying parameters of the infrastructure as workload input size and numbers of workers of the cluster, a model of the execution time can be obtained, that can then be applied to predict the execution time for different values of the same variables. The matemathical model is assessed and confirmed with later experimentation.

1. Introduction

In April 2013, a private company, located in Brazil, asked the GPPD/UFRGS laboratory for help to develop and test a Map Reduce application to process large amounts of logs. The logs come from their clients hourly, and have to be processed to feed a Database. The application is meant to be run in a Cloud Copmuting infrastructure, which can be briefly explained as one where the resources are configurable, accesible from the internet and paid by the time they are used [9]. Therefore, it is important to know the minimum cluster configuration in terms of quantity of nodes that can meet the demand of this application, and the time that will take to accomplish said demand. The goal for the analysis and help from the GPPD/UFRGS laboratory in this research was to study the behavior of the Map Reduce application as a part of a research that is currently being performed with MapReduce applications over the Cloud. We based our work on a procedure described in [4], which explained a simple but general methodology for modelling the performance of a parallel aplication running in a Cloud infrastructure, containing 4 steps as follows:

- Defining the parameters that affect the performance of a computer system;
- Perform observations of the application varying the parameters defined in the previous step and taking measures of the performance;
- Propose a mathematical model that relates the performance measures taken with the values of the parameters defined previously, and finally;
- Assess the developed model and test its accuracy.

The remainder of this paper is organized as follows: Section 2 presents the Motivation of this work; Section 3 describes how the experiments were conducted and its results; and finally, Section 4 states the conclusions and future work of this research.

2. Motivation

2.1. Map Reduce

Map Reduce is a programming model used to easily implement distributed programs. It was proposed by Jeffrey Dean and Sanjay Ghemawat in [5] from Google, and several implementations exist. MapReduce applications have a wide range of domains where they can be useful, and that is the reason for research to be done in order to expand it to add more features and correcting some of its features. The Hadoop [2] implementation of Map Reduce was chosen for this paper. The Map Reduce model lets the programmer define his Map and Reduce functions, however, it is restrictive enough so that the application can be automatically parallelized, scheduled and scaled on different machines, which could be physical or virtual.

A Map Reduce application is run in two phases:

- the Map phase analyses the input data and produces intermediate key/value pairs;
- the Reduce phase collects the set of intermediate keys with their associated values and merge them to form the output, also with a key/value format.

The Map and Reduce functions are written by the user and can easily be parallelized. For the Map phase, each machine analyses a part of the input, which is divided in pieces called *chunks*. And for the Reduce phase each machine processes a subset of the intermediate results arranged by key.

For our problem Map Reduce was chosen because of its ease of use and good performances for processing a lot of data.

2.2. Capacity Planning, Performance evaluation and Performance prediction

Capacity Planning is described in [8] as a process where it can be ensured that adequate computer resources will be available to the users of the system to meet future workload demands, and also [10] refers to Capacity Planning, defining it as properly design and size a computer system for a given load condition.

In [1], authors present the importance of having Performance models. The developed Performance Model is an analytical model that capture aspects of the system and relate each one by mathematical formulas and/or computational algorithms.

An interesting work in the topic of Map Reduce and Performance Prediction is [11]. In this paper, authors present a cost function that shows a relationship between some characteristics of the Map Reduce application and the time that takes for the application to execute; and also that they present also cost functions for running MapReduce applications on virtual machines in Cloud environments.

In [7], authors develop a way for performing Capacity Planning in Cloud computing environments for Map Reduce applications. Authors deal with the problem of determining the virtual machine cluster resources and the Map Reduce configurations to achieve user-defined requirements on execution time and economic cost for a given workload.

2.3. Goal

The goal of this paper is to develop a simple model to predict the performance in terms of execution time of a *log serialization application* running in a private cluster environment.

3. Experiments

3.0.1. Application As it can be seen in Fig. 1, the developed Map Reduce application receives files containing logs and transforms this input data in a serialized format so the data can then be more easily processed by a Database.

The input data files contain lines of a CDN server logs in a text format. Each file contains an hour worth of logs. Later, output files are classified in directories representing the date of the logs. These files contain all the serialized logs for each server, corresponding to the input files.

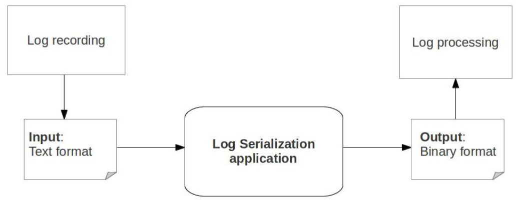

Figure 1. Block diagram of the application

3.0.2. Performance evaluation To make observations to the application and take measures of performance in terms of execution time, two parameters will be taken into account:

- input files sizes (W - workload)
  will be tested with values: $U_W = \{1, 5, 10, 20, 25\}$ GB, and
- number of machines that compose the cluster (*workers*) (p)
  will be tested with values: $U_p = \{4, 8, 12, 16\}$.

The goal of these measures is to find a function f which can gives us an expression of the execution time t of the application in terms of the number of workers p and the input size workload W: $t = f(W, p)$.

3.1. Experiment description

The set of tested parameters were all the combinations of the elements of U_W and U_p. Every combination of parameters was tested 10 times. The Map Reduce application was run on a cluster composed of 18 nodes in total. The cluster is located in INF/UFRGS and accesible at `gradep.inf.ufrgs.br` and each node has an Intel Pentium 4 2.79 GHz CPU and 2 GB in RAM.

Each time the application was run, the slaves were randomly chosen among these 18 machines to avoid errors produced when running every time in the same cluster nodes. Also, each time the application was run, the values of the parameters were randomly chosen. For example, after one run of the application with randomly chosen values of $W =$ 20 GB and $p = 4$ workers, the next run of the application was with randomly chosen values of $W = 5$ GB and $p = 16$ workers, and so on.

The Apache recommendations in [3] were used to select an optimal number of Reduce tasks. The number of reduces (N_R) followed this rule: $N_R = p \times n_{CPU} \times 1.75$. Where p is the number of workers, n_{CPU} is the number of CPU cores in the workers, and 1.75 is a value given by Apache.

3.2. Results

Results are shown in Fig. 2, where we can see that for a fixed numer of workers, the execution time increases along with the input workload size.

3.3. Mathematical model

The results were processed with R [6] to complete a linear regression. We used the model:

$$t_{exec} = \beta_0 + \beta_1 W + \beta_2 p + \beta_3 \cdot \frac{1}{p} + \beta_4 W \cdot p + \beta_5 W \cdot \frac{1}{p}$$

Where t_{exec} is the execution time we try to predict, W is the input size in GB, p is the number of workers and the β_i are the coefficients that the linear regression will calculate.

The linear regression gave:

$$\beta_0 = 465.8, \beta_1 = 16.8, \beta_2 = -11.6$$

$$\beta_3 = -714.9, \beta_4 = 2.1, \beta_5 = 569.8$$

So, β_0 represents the bias, the time that takes this application when it runs, indenpendent of the number of workers or the amount of input data; β_1 represents the time that takes for this application to schedule each GB of input data among its workers; β_2 represents the time that can be gained when adding one worker node to the cluster; β_3 also refers to the gained time when adding a worker node, but we can see that each time a worker node is added to the cluster, it makes the application gain a lesser amount of time; β_4 represents the time that the complexity of the cluster and the input data adds to the overall execution time; and finally, β_5 represents the time that the application gains per added node per GB of input data.

This model has a coefficient of determination R^2=0.99, which gives us a strong confidence in it.

3.4. Assessing the Model

After the mathematical model was proposed, it had to be assessed with experimental results. The same infrastructure of the experiments of section 3.1 was used, also the same values of *W* were used as well, being *W*={*1, 5, 10, 20, 25*} GB. The new values to assess the model were for *p*, being *p*={*6, 10, 14*} nodes.

So, the application was run in a cluster of *p* nodes to analyse data of size *W*. Experiments in this part were run 10 times, and the average values are described in Table 1.

For the results shown in Table 1 we can see that actually the model was good enough to predict the execution time for the conditions expressed in the Values column, with an error no bigger than 14% of the predicted value. With this results we say that the model is accurate and valid for execution time predictions with this application.

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%

Table 1. Results for model assessing

4. Conclusions and Future Work

Based on the results of section 3.4, we see that a mathematical model can be obtained and applied to predict the execution time of a Map Reduce application.

Using the methodology described in [4] helped to model the behavior of a parallel application in a private cluster environment. So, a future work will be to obtain a mathematical model of the same application when running in the Cloud, and will confirm (or not) if the methodology used in this work is general enough to help defining a mathematical model for other kinds of Map Reduce applications.

ACKNOWLEDGEMENTS

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

References

[1] V. A. Almeida and D. A. Menascé. Capacity planning an essential tool for managing web services. *IT professional*, 4(4):33–38, 2002.

[2] Apache, 2013. Welcome to Apache Hadoop! `http://hadoop.apache.org/` acessed on 24/07/2013.

[3] Apache, 2013. HowManyMapsAndReduces - Hadoop Wiki `http://wiki.apache.org/hadoop/HowManyMapsAndReduces` acessed on 24/07/2013.

[4] 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*

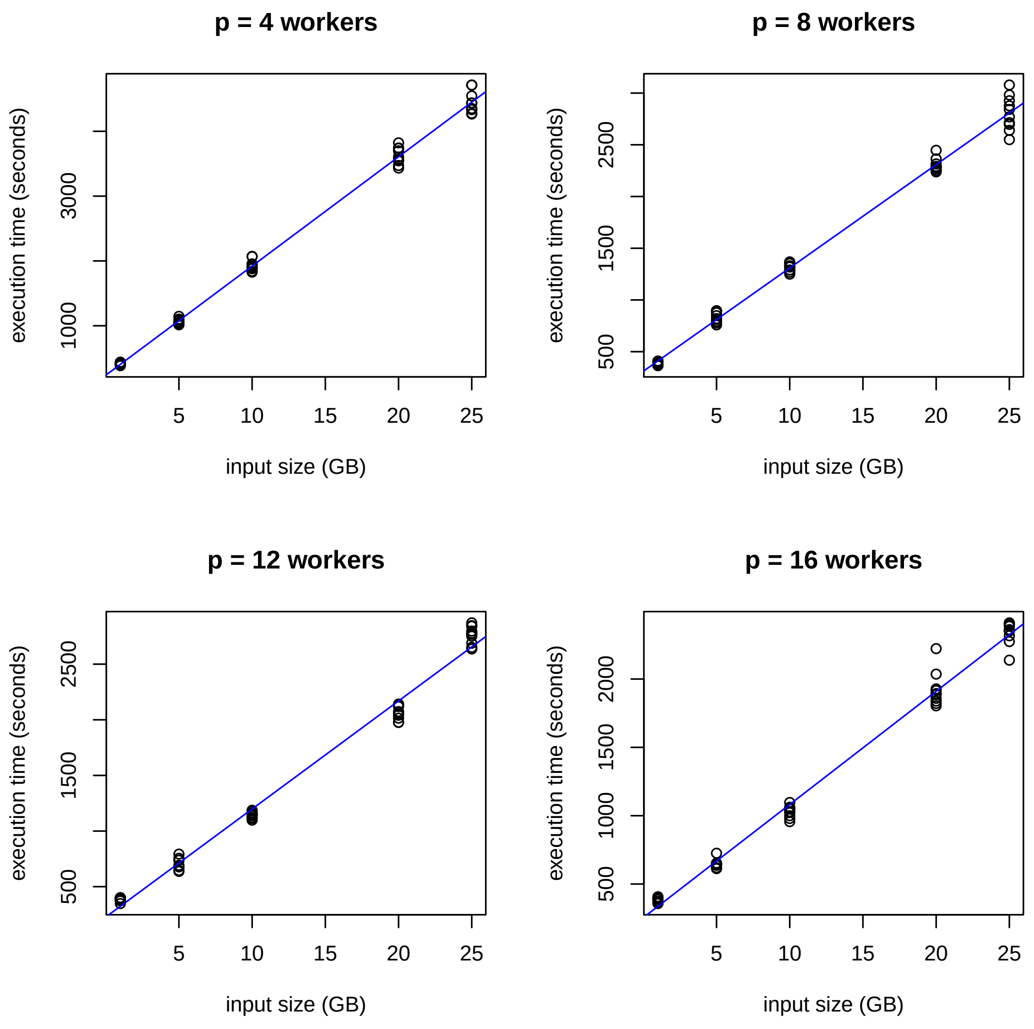

Figure 2. Results of execution time (in seconds) against input size of the workload (in GB) of the experiments, with indicated number of workers running the Map Reduce application

Conference on Enterprise Information Systems ICEIS - Vol. 2, pages 333 – 338. INSTICC, 2013.

[5] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] R. Foundation, 2013. The R Project for Statistical Computing `http://www.r-project.org/` acessed on 24/07/2013.

[7] 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*, page 18. ACM, 2011.

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

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

[10] D. A. Menascâe, V. A. Almeida, L. W. Dowdy, and L. Dowdy. *Performance by design: computer capacity planning by example*. Prentice Hall Professional, 2004.

[11] 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*, pages 155–162. IEEE, 2011.