# Prediction of Application Performance in Cloud Environments

Ciobanu Emilia
Tabara Mihai
Vatamanu Daniela

## *State Of The Art*

### 1. Introduction

In the past few years, distributed systems and especially Cloud architectures gained more and more popularity among various classes of users, thus becoming a robust, easy-to-scale and efficient solution for wide range of applications. Latest research address the problem of developing a central resource management system capable both of predicting how these resources vary over time and of dividing such resources in a more realistic and scope-centered way. This paper focuses on some of the most common components: the CPU and memory usage.

The main purpose of a RMSs (Resource Management Systems) is to log the performance of the resources in a grid, for future references, reviews, performance grading, parameter management, etc. While trying to solve scheduling on distributed systems is an NP-hard problem because you cannot foresee how systems will be asked to do tasks and how much computational bound they will be, it is certainly useful to try and find heuristics that will predict that within reasonable limits for given systems.

CPU prediction load is vital in task scheduling over distributed systems applications. Knowing when a host's current task will end can be of great help and will significantly improve efficiency of planning and assigning tasks amongst a number of processors. In delivering computing power as a service, one would like to offer their clients the best there is and that also means not having to wait for a long time. Assigning the task to the machine that will be free the soonest possible will greatly enhance the odds of clients using the services.

There are three main approaches that have been followed so far. All of them rely on taking sample data from the CPU in case, studying part of the data and, based on it, trying to predict the remaining sample data. The first approach involves linear time series models and computing the prediction set upon past patterns. Another direction will imitate the Markov chain AI model. This implies counting on defining a number of states for the processor and calculating the probabilities for the transitions between the states. Further algorithms rely on tendency, determining a slope in the latest CPU behaviour and figuring what the future load will look like based on the premise that it will follow the same slope.

Such methods greatly enhance task scheduling across distributed systems, as grids and clouds. The possibility of predicting load behaviour in a precise manner has become of major importance in the last years, in the race of building better structures and offering clients the best services available.

## 2. Prediction based on linear time series models

Linear time series models are used in many CPU load prediction algorithms. The reason these schemes are encountered so much is that the distributed systems in which they are implemented have certain patterns they follow. Studying the patterns they fall into seems to be an easy task that offers accurate enough results.

In their 2003 *Run-time Prediction of Parallel Applications on Shared Environments* paperwork, Byoung-Dai Lee and Jennifer Schopf propose a run-time prediction technique that uses regression methods and filtering techniques to derive the application executing time without using standard performance models.

There are several AI ins and outs that provide a basis for the way their regression methods are being applied. Thus, the CPU load history is being splitted in two parts. By using filters to evaluate, some subsets are being drawn out to be later applied the regression methods. This way, certain relationships in the data subsets reveal. Similar to Kapadia et al [1] and Lee and Weissman [2], their work is fully functional in dynamic environments showing an unexpected good behaviour, while very limited performance information is taken into consideration.

An N-body simulation code and a heat distribution code are the two parallel applications taken as subjects. Moreover, homogeneous and heterogeneous background loads are being used to complete the use case.

Their paperwork focuses more on the N-Body problem, which is concerned with determining the effects of forces between different objects, bodies or other entities (e.g. astronomical entities that are attracted for each other due to gravitational forces).The problem has similar interpretations and connotations in other fields such as fluid dynamics and molecular dynamics. Based on the master-slave paradigm, a quadratic solution is being implemented for the N-body problem which, afterwards, is being deployed on a 20 ~1 Ghz Intel Pentium III data-center where the resources (CPU load, bandwidth and latency are measured every 5 minutes).

Consequently, it is being concluded that the run-time predictors that take into account both CPU load and latency always produce better prediction accuracy(NP_PARM and DP among the best two predictors). The empirical evidence they achieve shows that the use of regression models delivers tolerable prediction accuracy. Moreover, the results can be dramatically improved by using appropriate filters.

Another overview on the scheduling in large scale distributed systems like Grids has been made by Hui Li and Lex Wolters from Leiden University in their paperwork, *An investigation of Grid Performance Predictions Through Statistical Learning*.

In their work, the authors first analyze the difference between local and Grid scheduling predictions, by emphasizing the backmost lack of control in terms of resources. As a counterpart for the old-fashioned static information such as machine types, number of processors and storage capacity, which can be obtained from monitoring the services, the authors propose for investigation some more dynamic type of information, thus application run time (how long a job will execute on the resource) and queue wait time (how long a job waits in the queue before starting). In order to achieve good results based on these two metrics, monitoring combined with historical data records analysis are being used.

A commonly known method of solving the problem is by using a ‘template of attributes’ to categorize jobs along with statistical techniques applied to generate predictions[3].

Instance Based Learning (IBL) is being used in [4] to investigate run-time predictions. Such techniques make use of historical data ‘near’ the query point to build a local model for approximation. In [5] scheduling algorithms like FCFS, LWF and backfilling are simulated to obtain predictions for queue wait times. In the same time, the authors draw attention to the drawbacks a simulation approach can have, in terms of both lack of scalability and impossibility to meet the real time requirements of the Grid. As an alternative, Li and Wolters, derive both application run times and queue wait times from historical observations. By combining these two, a so called ‘job response time’ on computing resources can be obtained.

IBL as a framework to predict both metrics, Bias-variance analysis of error and an efficient search tree structure called ‘M-Tree’ (used for better efficiency in lookup for the k-nearest neighbor search) are the tools used by the authors to achieve their goal.

The empirical evidence proves that acceptable prediction accuracy is achieved using the proposed techniques and the performance is able to meet the real time requirements in the Grid scheduling process.

Dinda et al. have studied in 1998, in *An Evaluation of Linear Models for Host Load Prediction*[6], the CPU load prediction through multiple linear models on Digital Unix. Box-Jenkins linear time series models such as AR, MA, ARMA and ARIMA have been proved to be relevant to reducing “confidence intervals”, which would heighten the job of selecting the best system for the job. The study findings were that autoregressive methods, AR, had the most accurate predictions amongst all of the examined algorithms.

In 1999, Dinda and O'Hallaron's *An Extensible Toolkit for Resource Prediction In Distributed Systems*[7] exhibit the design, implementation and performance of a toolkit used for online and off-line prediction of system loading. The systems they studied were independent and represented as streams of scalar measurements tested periodically. The outcome obtained from this project was rated as reasonably good and feasible for real-time systems.

In the following research, in 2001, in *Online Prediction of the Running Time of Tasks*[8], Dinda presents an algorithm for determining the running time of tasks which can compute valid confidence intervals. Through the use of such a method, coherent decisions can be made for distributing jobs on systems, thus being a useful and viable method to determine running times. The algorithm is based upon AR and is shown to be both cheap and easily implementable and scale to the systems it needs to model and schedule upon.

Vazhkudai, Schopf and Foster studied in *Predicting the Performance of Wide Area Data Transfers*[9] present their predicting framework which is meant for accurately predicting the necessary time needed to copy a large file from a cloud to a client. Although the task they assumed is quite difficult since the distance over which the file might have to be transferred can vary greatly, as well as network load and the load of the network devices found across the path followed from the server to the client.

The experiments were conducted on files of over 100MB and resulted in having a far better predictability for large files, rather than for small ones, as well as concluding that ARIMA models show no improvement in such cases with regard to other predictors. Median predictors have been shown to vary greatly, while predictors that proved to have the best outcome would then have significant misses, thus being no more efficient than those without great peaks.

## 3. **Prediction based on Markov chains**

Prediction based Markov chains have been shown to be effective in resource load predictions and have evolved from rudimentary 2-state system to multiple state modelling and now regard additional factors such as network traffic, number of jobs a system can perform without damaging execution time.

Lili at al. study in *A Markov Chain Based Resource Prediction in Computational Grid*[10] how representing the CPU load log in the form of Markov chain models can be both efficiently implemented and accurately predictive of future load of the resource.

The experiments are based upon the fact that resource load is similar to the previous days. Also, it relies on a set of 5 states the resources will be found in, according to how the network is being used at a particular moment. The factors that are thought to contribute to the load of a CPU are the number of compute-bound tasks assigned to it, amount of data being transferred across the network and reaching execution failure within the resource on behalf of excessive usage. Based on past information, we can define the states and the probability of transitioning between them.

First and foremost, updating the data must be considered and defining how this process works. Secondly, prediction must be measured and evaluated against the real values. The scheduling improvements will show mainly on the idle rate values, as it would mean better scheduling, which did happen in the experiments.

## 4. **Prediction based on tendency (slope method)**

Another method for determining future CPU load is through polynomial fitting method. Smooth and monotonous variance in CPU usage can be successfully modeled through polynomial curves. Although it has a downside, in that it cannot foresee turning points this way, thus resulting in faulty data until it converges again, it will keep a close watch on actual data behaviour otherwise.

Zhang, Sun and Inoguchi explore a possibility similar to this in *CPU Load Predictions on the Computational Grid*[11]. The patterns they observed would follow in the lead of either past steps, or follow analogous past patterns, providing through a hybrid model, much better results.

## *Bibliography*

[1]: Nirav H. Kapadia, Jose A. B. Fortes, and Carla E. Brodley, Predictive Application-Performance Modeling in a Computational Grid Environment, 1999.

[2]: Byoung-Dai Lee and Jon B. Weissman, Adaptive Resource Scheduling for Network Services, 2002.

[3]: W. Smith, I. Foster, and V. Taylor. Predicting application run times using historical information, 1998.

[4]: N. H. Kapadia, J. A. B. Fortes, and C. E. Brodley. Predictive application-performance modeling in a computational grid environment, 1999.

[5] W. Smith, V. Taylor, and I. Foster. Using run-time predictions to estimate queue wait times and improve scheduler performance, 1999.

[6] P. A. Dinda and D. R. O'Hallaron. An Evaluation of Linear Models for Host Load Prediction, 1998.

[7] P. A. Dinda and D. R. O'Hallaron. An Extensible Toolkit for Resource Prediction In Distributed Systems, 1999.

[8] P. A. Dinda. Online Prediction of the Running Time of Tasks, 2001.

[9] S. Vazhkudai, J. M. Schopf and I. Foster. Predicting the Performance of Wide Area Data Transfers, 2001.

[10] S. Lili, Y. Shoubao, G. Liangmin, W. Bin. A Markov Chain Based Resource Prediction in Computational Grid, 2009.

[11] Y. Zhang, W. Sun, Y. Inoguchi. CPU Load Predictions on the Computational Grid, 2006.