A FRAMEWORK FOR PERFORMANCE PREDICTION IN DISTRIBUTED SYSTEMS

Corina STRATAN[1], Valentin CRISTEA[2]

În sistemele distribuite de dimensiuni mari, parametrii de performanță ce caracterizează diversele componente variază frecvent, făcând dificilă gestiunea resurselor. În această lucrare prezentăm o platformă pentru predicția performanțelor, ce poate fi utilizată pentru a îmbunătăți eficiența sistemelor de gestiune a resurselor. Platforma este integrată în sistemul de monitorizare MonALISA şi este flexibilă, permițând selecarea unuia dintre algorimtii de predicție disponibili sau adăugarea unor algoritmi noi. Am testat platforma în cadrul rețelei USLHCNet, pentru predicția traficului în rețea.

In large scale distributed systems, the performance parameters of the various components change frequently, making resource management a challenging task. In this paper we present a performance prediction platform that can be used to improve the efficiency of resource management systems. The framework is integrated in the MonALISA monitoring system and is flexible, allowing the user to select among the available prediction algorithms or to add new ones. We have tested the framework by forecasting network traffic in the USLHCNet network.

Keywords: prediction, distributed systems, performance, monitoring

1. Introduction

One of the major challenges in developing applications for large scale distributed systems is their dynamism. Due to the complexity of the systems and to the large number of users, it is highly probable for some of their characteristics to change even in a short time period. Changes may occur at various levels (hardware, OS, application etc.), and there are different strategies that can be used to handle them. We shall focus here on the changes that occur in the workload of the distributed resources (CPUs, network links, memory etc.).

Besides the fact that the utilization of the distributed resources varies frequently, another problem is that existing resources may fail or new ones may join the system. These problems have been identifed as soon as large scale distributed systems became a viable alternative to supercomputers; one of the first

[1] Prof., Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, corina.stratan@cs.pub.ro
[2] Assist., Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania

strategies to tackle it (introduced in several works, like [1]) was dynamic resource management. Unlike the static approach, in the dynamic one the resource allocation is adjusted "on the fly", as a consequence to the variations in the system's workload and also in in the users' demands.

Dynamic resource management often relies on prediction mechanisms in order to get an estimation of what the resources utilization will be in the near future, or of how many resources are likely to fail. For this reason, a prediction mechanism constitutes an important component of any resource management system. In this paper we present a framework for performance prediction integrated in the MonALISA monitoring platform [2]. The framework is flexible, allowing the users to select among several available prediction algorithms and also to add new algorithms.

In the folowing paragraphs we make a short overview of the most significant projects related to performance prediction in distributed systems.

One of the most well-known performance prediction frameworks for distributed systems, the Network Weather Service [3], has been used in several scheduling projects to improve the mechanisms of assigning tasks to computational resources. The Network Weather Service has a distributed architecture, comprising sensor elements that gather monitoring information, memory elements that store the data as time series, and nameservers that facilitate the discovery and the communication among elements.

One of the scheduling projects that uses the Network Weather Service is AppLeS (Application Level Scheduler) [4]. AppLeS uses monitoring and benchmarking to obtain information about the current performance of Grid resources in a dynamic way; the results are further used in meta-scheduling. The jobs are submitted with the aid of a broker (or navigator), together with their hardware and software requirements; the broker then consults the GridScape, which is a description of the Grid resources and of their current state. The GridScape is updated with the results obtained by running periodically the NAS Grid Benchmarks [5], and also with information provided by various monitoring tools among which is the Network Weather Service.

[6] presents another approach for using prediction when making scheduling decisions. The approach is based on a real-time scheduling advisor (RTSA), which performs a statistical time series analysis in order to recommend execution hosts for the tasks and also to predict their running time.

Another area of use for prediction is related to resource availability; one of the most recent works in this direction is [7], which introduces a system for resource availability monitoring, analysis, and prediction, named GriS-Prophet. Knowing the probability for certain resources to be available in the near future can significantly improve scheduling decisions; GriS-Prophet addresses this need by

providing resource availability prediction. The project also introduces a classification of Grid resources on the base of their availability characteristics.

Regarding the prediction methods, among the most widely used are the moving average techniques (that use a set of past parameter values from a time series in order to predict the next value, by calculating the average of the past values or by applying a more complex formula). The Network Weather Service [3] uses several moving average methods, and automatically chooses from them the method that is likely to be the most accurate at a given time step. Neural networks are also frequently used for prediction, especially regarding network measurements; a study of neural network predictors is done in [8]. [7] applies methods from pattern recognition and classification, like Bayesian Inference and Nearest Neighbor Predictor.

In the following sections we introduce our approach for performance prediction, by presenting the general architecture of the prediction framework, the prediction methods that we have used and some experimental results.

2. The Architecture of the Prediction Framework

Among the main design goals of the prediction framework we have developed were the scalability, the flexibility and the ease-of-use. These goals motivated the way we have structured the framework, by separating it into a front-end that is integrated in the MonALISA repository and has the role of interacting directly with the user, and a back-end that executes the actual prediction algorithms. The MonALISA repositories are databases that store monitoring information for long periods of time, and also provide a web interface based on Java servlets, which allows the visualization of the stored data.

We shall briefly introduce as follows the components of the prediction framework, that are presented in Fig. 1.

The web interface, based on Java servlets, is integrated with MonALISA repository's interface and allows the user to:

- conFig. prediction parameters (the prediction method, the learning interval and other specific parameters)
- visualize the prediction results, together with the real values

We have augmented the MonALISA repository's database so that it stores predicted values, together with the actual parameter values collected from the monitored sites. For a better flexibility, the predicted values are introduced in separate tables and can be stored on a long term; this allows a further analysis of these values and of their accuracy. The predicted values are inserted into the

database with the aid of the prediction connector, which acts as an interface to the prediction back-end. The prediction connector periodically sends parameter values to the prediction engine; after applying a prediction method, the engine sends back predicted values to the prediction connector, which inserts them into the database. The prediction connector also transmits the settings chosen by the user (prediction algorithm, learning interval etc.) to the back-end.

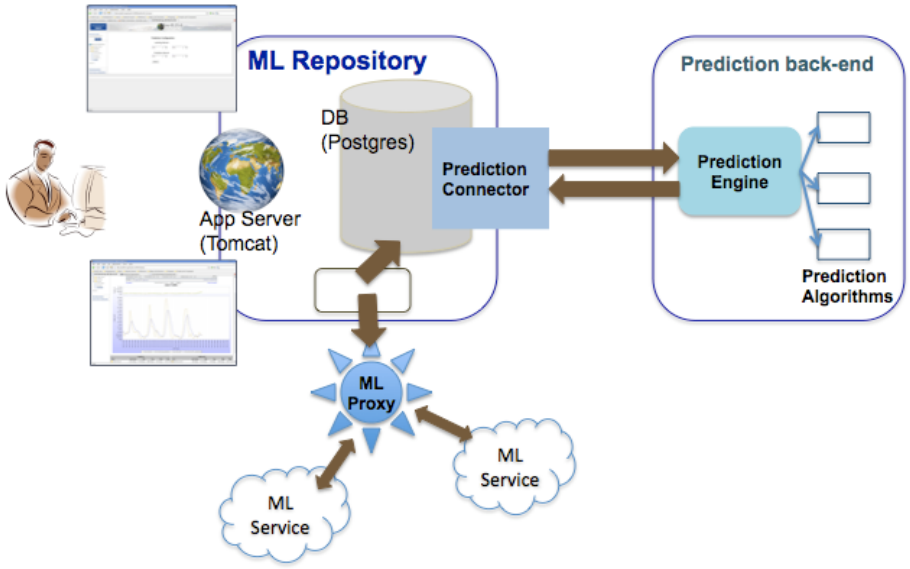

Fig. 1. The architecture of the MonALISA prediction framework.

The prediction back-end is separated from the repository for scalability reasons; thus, it will be able to perform complex prediction algorithms without introducing an additional load on the repository's machine (which is heavily loaded most of the time with client requests). Also, the prediction algorithms are developed separately from the prediction engine, so that we have the possibility to introduce new algorithms with a minimal effort. So far we have implemented three prediction methods that we present in the next section: moving average, weighted moving average and exponential smoothing.

3. Prediction Methods

We have implemented in the framework three prediction methods of the "moving average" type, that we shall present in this section; the methods are presented in more detail in [9].

The prediction methods take as input a set of k parameter values from a time series, denoted as: x_t, x_{t-1}, …, x_{t-k+1}. These are values obtained from monitoring the parameter, which can be any parameter from a distributed system that has a numerical representation. The methods aim to obtain a forecast for the value of the parameter at the next time step, x_{t+1}. We shall denote the forecasted value with $xpred_{t+1}$. As a simplification, we denoted the time steps with consecutive natural numbers; in reality, the measurements are done at fixed time intervals that have a length greater than 1, but the simplification does not affect the prediction algorithm.

Moving Average

The moving average method calculates a "simple" average of the last k observations and assigns this value to the parameter forecast:

$$xpred_{t+1} = \frac{x_t + x_{t-1} + ... + x_{t-k+1}}{k} \quad (1)$$

This method has the advantage of being extremely simple to perform, consuming minimal computational resources. On the other hand, the produced values might be less accurate, as all the past values have an equal importance when calculating the forecast. If the parameter varies rapidly and k is large, the method gives too much importance to values obtained a long time before and produces an inaccurate forecast (the curve for the forecasted values will be much smoother than the one for the real values). Another disadvantage of this method is that it can be applied only after at least *k* measurements have been taken.

Weighted Moving Average

This method derives from the moving average method, but aims to improve it by providing the possibility to assign different weights to the values observed in the past. In this way we can give a greater importance to the most recent values, by assigning them larger weights (and this is the most common practice for using this method).

The forecasted value is calculated with the following formula:

$$xpred_{t+1} = \frac{w_1 \times x_t + w_2 \times x_{t-1} + ... + w_k \times x_{t-k+1}}{k} \quad (2)$$

We have implemented this method by assigning to the most recent three values a double weight compared with the other values. As we shall show in the

next section, we have obtained better results than with the simple moving average method.

Exponential Smoothing

The exponential smoothing method is similar with the weighted average, assigning greater weights to the most recent observation and smaller weights to the older ones. Specifically, the forecasted values are calculated as follows:

$$\begin{aligned} xpred_0 &= x_0 \\ xpred_{t+1} &= \alpha \times x_t + (1-\alpha) \times xpred_{t-1} \end{aligned} \tag{3}$$

where is the smoothing factor, and is a real number in the (0, 1) interval.

By repeatedly substituting the values of xpred into this equation, we obtain a sum in which the weights for the past observations of x are in geometrical progression: 1, (1-α), $(1-\alpha)^2$, $(1-\alpha)^3$, … Small values of α give more importance to the older observation, and have a strong smoothing effect, while larger (close to 1) values give more importance to the recent observations.

4. Experimental Results

In order to test the prediction framework, we have conFig.d a MonALISA repository to collect and store monitoring information from the US LHCNet network [10]. LHCNet provides high-performance transatlantic connections between several Tier 0 and Tier 1 computing centers from the LHC experiment. With 10 Gbps bandwidth in its links, the network has been successfully used for a wide variety of tasks, especially large file transfers.

The parameter for which we have tested the prediction methods was the network traffic between two computing centers, one placed at CERN and the other one placed in Chicago. More specifically, we used two parameters, one representing the traffic from CERN to Chicago and the other one representing the traffic from Chicago to CERN.

We have applied the following prediction methods for time intervals of 12 hours each:

- moving average
- weighted average
- exponential smoothing with 10% smoothing factor
- exponential smoothing with 20% smoothing factor

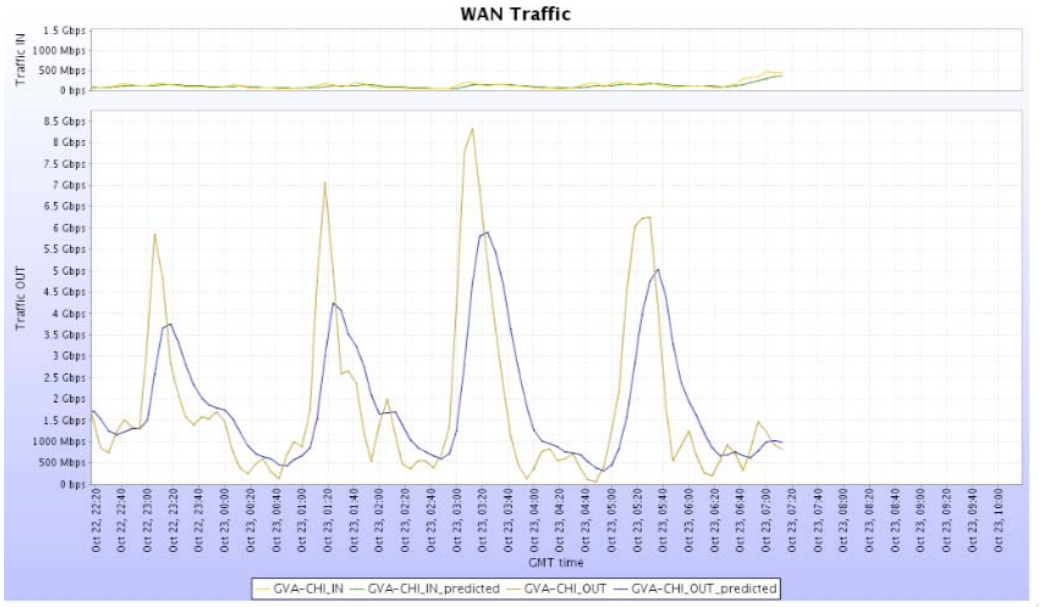

Fig. 2. Prediction results for the exponential smoothing method

Fig. 2 shows the prediction results for the exponential smoothing method with a 20% smoothing factor. The upper plot represents the traffic from Chicago to CERN, and the one below it represents the traffic from CERN to Chicago. Both plots represent the measured traffic values (in yellow) and the predicted values (in blue and green). As we can see, the predictions are less accurate when the parameter values vary rapidly (which happens in the lower chart).

Fig. 3 presents the prediction results for the moving average method; by comparing it with Fig. 2 it is clear that the exponential smoothing method has a better accuracy.

In order to compare quantitatively the accuracy of the prediction methods, we have computed for all of them the mean squared error and the average error. The mean squared error (MSE) represents the average of the square of the error for a prediction method (the error being the difference between the measured value and the predicted value); it is calculated as follows:

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (xpred_i - x_i)^2 \tag{4}$$

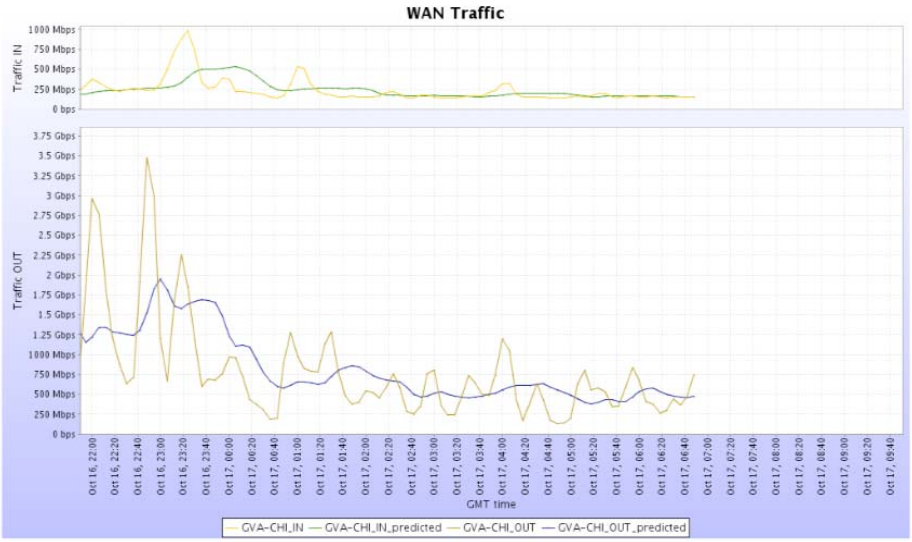

Fig. 3. Prediction results for the moving average method

We have also computed the average error, which we denote by AVGERR, as the average deviation of the predicted values from the real values:

$$AVGERR = \frac{1}{n} \times \sum_{i=1}^{n} | xpred_i - x_i | \tag{5}$$

The values are presented in Table 1, and were obtained from measurements of the network traffic from Chicago to CERN. The table also shows the following metrics:

- Avg measured value (in Mbps): the average of the measured values in the time interval for which we applied the prediction method
- Std deviation: the standard deviation for the measured parameter
- Err percent: the percentage of the average measured value that is represented by the error

As the table shows, the predictions were not very accurate (the error percentage was around 20-30%), which can be explained both by the high variability of the data and by the simplicity of the prediction methods. We can also see from the results that the exponential smoothing method generated more accurate predictions.

Table 1

Accuracy estimations for the prediction methods

Method	Avg. measured value (Mbps)	Std. deviation	MSE	AVGERR	Err. percent
Moving avg.	232.35	135.28	126.22	75.04	32.29
Weighted avg.	56.65	28.91	27.09	17.81	31.44
Exp. smooth 10%	117.26	75.25	45.63	35.08	29.91
Exp. smooth 20%	276.12	145.04	125.16	55.94	20.26

5. Conclusions

Dynamic adaptation is an essential aspect in distributed systems, as in such systems changes occur frequently and at various levels. The type of changes that we have approached is the variation of the distributed resources' workload; one solution for adapting to this type of changes is to use a prediction mechanism in order to estimate the availability of the resources in the near future.

We have developed a flexible prediction framework integrated with the MonALISA monitoring system, and have used it to forecast the available bandwidth for links from the USLHCNet network. Within the framework we have implemented three moving average prediction algorithms, and in the future we intend to develop more complex methods that could provide better forecasts for longer periods of time (specifically, we plan to investigate methods based on neural networks and on pattern detection).

R E F E R E N C E S

[1] *J.E. Moreira, V.K. Nai*, "Dynamic resource management on distributed systems using reconfigurable applications", IBM Journal of Research and Development, **vol. 41**, 1997, pp. 303-330

[2] *Legrand, I., Grigoras, C., Muraru, A., Musat, L., Toarta, M. and Voicu, R.*, "Information gathering and management in MonaALISA framework", in Proceedings of the 15th International Conference on Control Systems and Computer Science (CSCS15), Bucharest, Romania, 2005

[3] *D.M. Swany, R. Wolski,* "Multivariate resource performance forecasting in the network weather service", in Proceedings of SC02, Baltimore, MD Nov. 2002, pp. 1-10

[4] *M. Frumkin, R. Hood,* "Using grid benchmarks for dynamic scheduling of grid applications", NAS Technical Report, 2003

[5] *M.A. Frumkin, der Wijngaart*, R.F.V. (2002), "NAS grid benchmarks: A tool for grid space exploration", in Cluster Computing no. 5, 2002, pp. 247-255

[6] *P.A. Dinda,* "A prediction-based real-time scheduling advisor", in IPDPS, IEEE Computer Society, 2002, pp. 35

[7] *F. Nadeem, R. Prodan, T. Fahringer, A. Iosup,* "A framework for resource availability characterization and on-line prediction in large scale computational grids", Tech. Rep. TR-

0130, Institute on Resource, Management and Scheduling, CoreGRID - Network of Excellence, 2008

[8] *R.J. Frank, N. Davey, S.P. Hunt,* "Input window size and neural network predictors", in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), **vol. 2**, 2000, pp. 237-242

[9] *** "NIST/SEMATECH e-Handbook of Statistical Methods" [Online] Available: http://www.itl.nist.gov/div898/handbook/

[10] *** US LHCNet web page. [Online] Available: http://lhcnet.caltech.edu/