Decomposition based Algorithm for State Prediction in Large Scale Distributed Systems

Mihai Istin, Andreea Vişan

Faculty of Automatic Control and Computers, Computer Science Department,
University "*Politehnica*" of Bucharest, Romania
Emails: mihai.istin@cti.pub.ro, andreea.visan@cti.pub.ro

***Abstract*—Prediction represents an important component of resource management, providing information about the future state, utilization and availability of resources. We propose a new prediction algorithm inspired from the decomposition of a complex wave into simpler waves with fixed frequencies (similar to Fourier decomposition). The partial results obtained from this decomposition stage are combined using approaches inspired from artificial intelligence models. The experimental results for different system parameters, used in Alice experiment, highlight the great improvement, discussed in terms of error reduction, offered by this new prediction algorithm. The tests were made using real-time monitoring data provided by a system monitoring tool, in the case of one-step and multi-step ahead prediction. The prediction's results can be used by the resource management systems in order to improve the scheduling decisions, assuring the load balancing and optimizing the resource utilization.**

Keywords

State Monitoring, Prediction, Neural Network, Fourier Decomposition, Distributed Systems

I. Introduction

The size of distributed systems followed an ascending trend during the last decade leading to today's Large Scale Distributed Systems (LSDSs). As a result, resource management became more and more important as a main component used for improving the performance of the system. Scheduling, meaning the assignment of received tasks to distributed computational resources, is one of the most important parts of resource management [4], [1].

Monitoring can contribute to the improvement of the functionalities and the performance of resource management systems in various ways, and for this reason most of the current resource management systems use monitoring instruments. The most important functional requirements that a monitoring module should satisfy are: obtaining accurate information for all relevant parameters, support for various data delivery models, extensible data representation, access to real-time and history data, intuitive results visualization, and portability. We have also to mention the scalability, a minimal monitoring overhead and the ability to process and distribute, in real-time, large amounts of gathered data.

Resource management often relies on prediction mechanisms in order to estimate what the resources utilization will be in the near future, or how many resources are likely to fail. The prediction mechanisms use monitoring data that have been collected from the system and usually stored in databases. Based on a set of previous values for a certain parameter, the next value is forecasted and sent to a resource manager. Classical prediction algorithms are based on mean, median and standard deviation theory, but can also be used approaches using natural models.

The paper proposes a new prediction algorithm inspired from the decomposition of a complex wave into simpler waves with fixed frequencies, similar to a Fourier decomposition. The partial results (obtained for each "frequency" represented by sampling step in our case) represent the inputs for a neural network structure constructed using artificial intelligence methods. Our experimental results highlight the great error reduction offered by this approach, compared to classical prediction algorithms in both one-step and multi-step ahead prediction situations. We have used a framework for distributed system monitoring, real time prediction evolution for the proposed algorithm and also a real-time error analyzer for other well-known approaches.

The paper is organized as follow: Section II presents the related work referring to prediction approaches. Section III briefly introduces the components of the monitoring and state prediction tool. Next, Section IV presents the new prediction algorithm. In Section V, a performance comparison between the new algorithm and classical prediction algorithm is made for different system parameters. The experimental tests were made in both one-step ahead and multi-step ahead prediction. The results highlight the great improvement obtained by our new approach. Section VI presents the conclusions.

II. Related Work

This section presents a brief description of other prediction algorithms. The correlation between prediction and monitoring is important because performance prediction and evaluation of LSDS resources and applications are used to implement for example adaptive scheduling and on-line fault detection.

Different techniques have been developed to predict the behavior of available resources, such as CPU load, network traffic, storage systems or memory. Performance predictions are made using compositional models based on previous behavior and static characteristics of the resources. Table I summarizes the proposed approaches.

TABLE I
DESCRIPTION OF PREDICTION ALGORITHMS

Name	Prediction Approach
Simple Moving Average (SMA)[3]	The un-weighted mean of the previous n data points
Weighted Moving Average (WMA) [3]	The weighted mean. The weights are decreasing arithmetically as the values are older in time.
Exponential Moving Average (EMA) [3]	Applies weighting factors which decrease exponentially.
Random prediction	Random number contained in the interval $(mean - standardDeviation; mean + standardDeviation)$
Backpropagation NN [10]	A multi-layer NN using delta-rule for corrections
CasCorAG NN [12]	NN architecture dynamically constructed, initiated using genetic algorithms

Different prediction algorithms have been proposed in the current literature [3], [10], [8], [9] based on median, mean and standard deviation. A moving average [3] is used to analyze a time series by creating a series of averages of different subsets of the full data set. It is a type of convolution and it is similar to the low-pass filter used in signal processing. Several methods of computing the next value have been developed: simple, weighted and exponential moving average.

The random predictions are based on the standard prediction theory. Standard deviation measures the spread of data about the mean. The random prediction algorithm computes one step ahead value as a random real number contained in the interval $(m - s, m + s)$ where m is the mean of the set of values and s is the standard deviation.

Other prediction algorithms based on neural networks (NNs) [6], [11] have been also implemented thanks to their learning ability from historical data and thanks to their capability of multi-step ahead prediction. In [12] we proposed a state prediction algorithm based on the Cascade Correlation NN architecture initialized using a genetic algorithm in order to obtain better results and fast performance. Combining the advantages of NNs with a dynamically constructed architecture, fully adapted to the characteristics of the time series, the algorithm proved to be able to provide more accurate results than classical prediction methods and the back-propagation prediction algorithm.

This paper proposes an optimization of our previous work. We consider a new approach in the evaluation of the time series that we want to predict, computing the next value by taking into account not only past values, but also older values. We first divide the current time series into multiple series using different sampling steps, than we compute the next value for each of them and finally, we compose the results using machine learning techniques.

III. THE MONITORING AND STATE PREDICTION TOOL

The main design goals of the monitoring and state prediction tool we have developed were the scalability, the flexibility and the ease-of-use. Figure 1 presents the architecture of the monitoring and state prediction tool, where the main components are: the monitoring module, the repository server, the prediction server, the database and the web server.

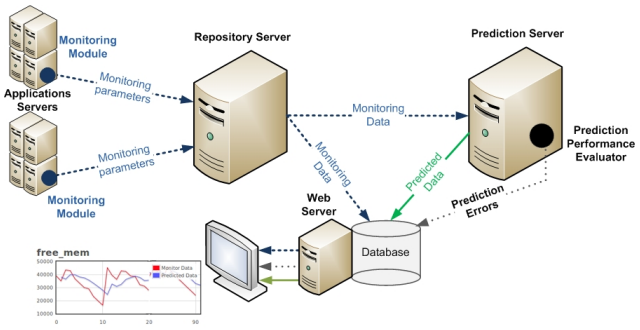

Fig. 1. The Architecture of The Monitoring and State Prediction Tool

A. The Monitoring Module

The aim of the monitoring module is to provide accurate information about the current values of system parameters, collecting them from system files. The most important functional requirements of a monitoring module are: extensibility, scalability, portability, intuitive results visualization, minimal monitoring overhead and the ability to process and distribute, in real-time, large amounts of monitoring data.

The collected parameters are grouped into two categories: dynamic system parameters and general (static) system parameters. The dynamic system parameters represent various aspects from the monitored systems that usually vary in time. We present below the list of dynamic system parameters considered by our monitoring tool:

- *cpu usr, cpu sys, cpu nice, cpu idle*: percent of the time spent by the CPU in user mode, in system mode, in nice mode, in idle mode
- *cpu usage*: CPU usage percent
- *load1, load5, load15*: average system load over the last minute, last 5 min and last 15 min
- *mem free, swap free*: amount of free memory and swap memory, in MB
- *number of total, blocked and running processes*
- *iowait, irq, softirq*: waiting time for IO operations, number of interrupt requests and software interrupt requests

The group of general (static) system parameters characterizes the system's characteristics that does not vary in time: the machine's hostname, model name, CPU frequency, number of CPUs and total cache size.

At the base of the monitoring service is a multithreaded system that independently executes various data collection

tasks in parallel. Each thread is responsible for dynamically loading and executing the modules that collect different sets of information. The threads initially created are reused when a task assigned to a thread is completed, in order to reduce the load on systems running MonALISA. Due to the independent threads that compose the system, a failure in one monitoring task will not affect the execution of other tasks. The tasks that need to be periodically executed are kept in a priority queue, thus, monitoring a large number of heterogeneous nodes with different response times can be done without difficulty.

B. The Repository Server

The repository module gathers monitoring information from different systems, forwarding them to a prediction server and also to a database who stores the history of each monitored station for a possible future data mining. The repository module implementation is based on the multi-threading paradigm.

The monitoring module works similar to a state machine (see Figure 2). Messages sent between the monitoring module and its clients have one of the following types:

- *authentication message* - contains the static parameters and represents a connection request made by a station to a repository server.
- *acknowledgment message* - contains the secret key offered by the repository server to the station in case of acceptance.
- *registration message* - contains the previously obtained secret key and represents a reconnection request made by a station that tries to reconnect to the same repository server.
- *done message* - contains the reconnection acknowledgment sent by the repository server to the requester.
- *update message* - contains monitoring data, periodically sent to the repository server.

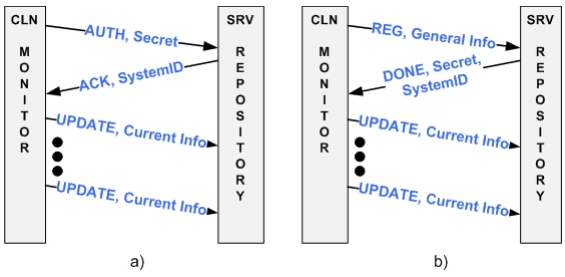

Fig. 2. Communication protocols a) with authentication b) with registration

C. The Prediction Module

The prediction module receives different monitoring data and predicts the future values of the parameters, for both one-step and multi-step ahead prediction. The predictions can be made using different algorithms such as: simple, exponential and weighted moving average, random prediction, CasCorGA prediction [12], etc and also prediction using the new approach proposed in this paper.

The prediction module also provides a performance evaluator useful to compare the results of different algorithms on the same time series in order to choose the fittest algorithm for each parameter.

As a performance criterion, we consider the Absolute Percentage Error (APE), that is computed using the following formula:

$$APE(p,r) = \sum_{i=0}^{n} \frac{|p_i - r_i|}{r_i}$$

where:

- r_i is the real value at the i^{th} measurement
- p_i is the predicted value for r_i
- n represents the number of measurements taken into account

D. The Web Server

The web server provides a flexible interface that offers a real time perspective of both gathered and predicted parameters. It gives the possibility to inspect the behavior of the monitored hosts, offering real time information of different parameters that characterize the system's state, utilization and availability. More, the web server provides an evaluation framework for the proposed prediction solution, comparing its performance with results of classical prediction approaches. the performances of different prediction algorithms are evaluated according to the previously described criterion.

IV. Decomposition Based Prediction Algorithm

All the prediction algorithms proposed in the literature take into account the last consecutive w values in order to predict the next future value of the time series that describes the resource's behavior. This approach is represented in Figure 4, where:

- w is the window size, meaning the number of history values used to predict the future values (in Figure 4, w is equal to five)
- $V(t)$ represents the measured value at the t^{th} measurement
- the goal is to predict the next future value, $V(t+1)$, marked in the figure with a filled bullet
- the marked values are those taken into account by the prediction algorithm.

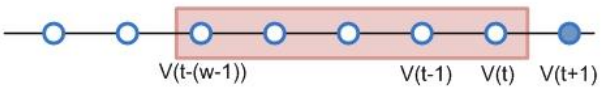

Fig. 4. Model of classical prediction approaches

Considering the prediction of the future value $V(t+1)$, denoted by $PV(t+1)$, the prediction in the classical approach is made according to the following formula:

$$PV(t+1) = f_{pred}(V(t-(w-1)),.,V(t-1),V(t))$$

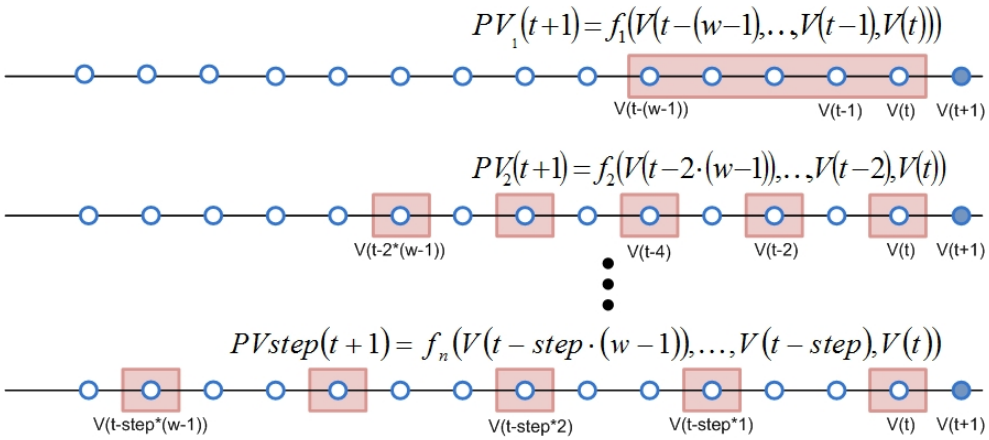

Fig. 3. New prediction approach

where f_{pred} is the used prediction algorithm and may be an algorithm based on mean, median, standard deviation or a complex one, inspired from artificial intelligence methods.

We propose a different prediction approach that takes into account not only the last w values as the classical solution previously described, but also considers older values with different sampling rates. This model is inspired from the decomposition of a complex wave into simpler waves with fixed frequencies.

We divide the time series that describes the current state of one resource, into multiple time series using different sampling steps. This represents in fact the decomposition stage presented in Figure 5. The number and values of sampling step should be carefully chosen. These issues are further discussed in the paper.

For each of the obtained series, we predict the value of $V(t+1)$ using one of the classical algorithms. Then, in the composition stage of the algorithm, all these prediction results are combined into a single value using methods inspired from artificial intelligence, namely neural networks.

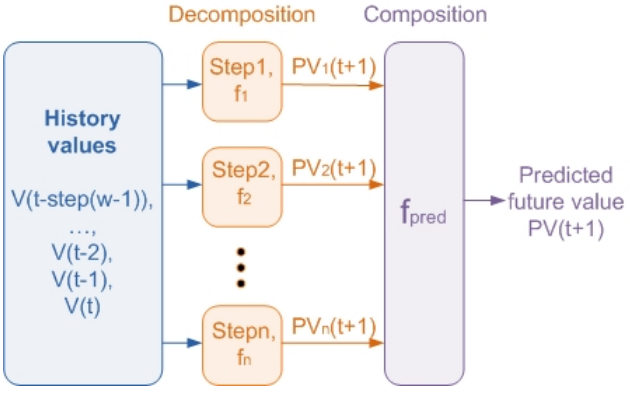

Fig. 5. New prediction approach based on wave decomposition followed by composition

Figure 3 presents the time series decomposition considered by our prediction algorithm, where:

- $f_1, f_2, ..., f_n$ are the prediction algorithms applied on the first, second, ... n^{th} sampling series, and can be one of the prediction algorithms named in Section II. f_i with $i \in 1..n$ computes PV_i, the predicted future value.
- $PV_1, PV_2, ..., PV_n$ are the predicted values of $V(t+1)$ of the first, second, ... n^{th} step

Different sampling steps can taken into account. Figure 3 presents an example that uses 3 different sampling step values (1, 2, 3) corresponding to PV_1, PV_2, PV_{step}. Customizing the new prediction approach and considering only one sampling step, equal to 1, we obtain the classical prediction approach, presented in Figure 4.

For each sampling step (denoted by i), the partial prediction is computed using the following formula:

$$PV_i(t+1) = f_{pred_i}(V(t-i(w-1)),.,V(t-i),V(t))$$

As previously mentioned, those partial results (PV_i with $i \in 1..step$) are further combined for the final computing of the predicted value for the moment $t+1$ according to the following formula:

$$PV(t+1) = f_{pred}(PV_1(t+1), PV_2(t+1), ..., PV_{step}(t+1))$$

In our experimental tests, we use three different sampling steps (one, two, three) in the decomposition stage (Figure 5), solution motivated further in the paper. In order to predict each branch (PV_1, PV_2, PV_3) we use a prediction algorithm based on average and in order to compose all those results (instead of f_{pred} in Figure 5) we use a learning algorithm based on NNs: a perceptron or a complex architecture inspired for the Cascade Correlation NN architecture [5], [7].

The first learning algorithm uses a perceptron which is the simplest kind of feed-forward NN: a linear classifier. We considered the architecture presented in Figure 6, containing three real inputs (corresponding to each partial prediction result - PV_1, PV_2, PV_3) and one real output and trained using the

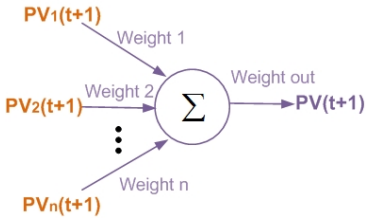

Fig. 6. The perceptron's architecture used in the composition stage

delta rule. The threshold is considered zero and accordingly, the output is computed using the following formula:

$$PV(t+1) = Weight_{output} \sum_{i=0}^{n} (PV_i(t+1) * Weight_i)$$

where:

$$Weight_{output} \sum_{i=0}^{n} Weight_i = 1.$$

The second learning algorithm uses a complex architecture inspired for the Cascade Correlation NN architecture [5], [7] (Figure 7, where the hidden layers are represented using a circle, the frozen weights are represented using a filled square and the trained weights are represented using an unfilled square). This kind of NN has a dynamical structure which adapts to the properties of the function to be learned by adding multiple hidden layers until the correlation between the input weights of those hidden layer and the previously obtained errors. Another important advantage of the Cascade Correlation NN is that is requires no back-propagation of error signals through the connections of the network.

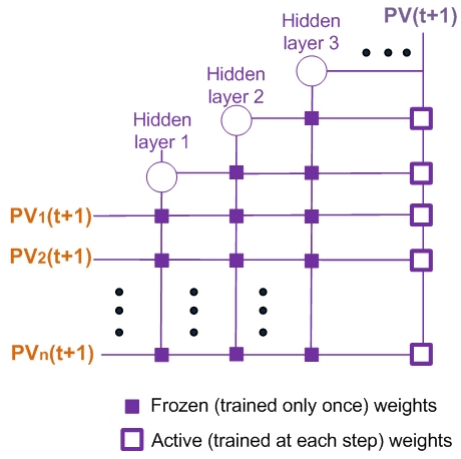

Fig. 7. The Cascade Correlation architecture [5], [7] used in the composition stage

The Cascade Correlation NN architecture is dynamically constructed in the following way [12]: the initial input weights are set according to a weighted moving average and the initial output weights are trained according to the delta rule, trying to minimize the errors for the considered training set. Until the errors are inacceptable we add a new hidden layer. Its input weights can be set for instance using the results of a genetic algorithm, trying to maximize the correlation between the current errors and those weights. Its output weight is trained together with the other output weights according to the same rule until the output of the neural network reaches a threshold of accuracy.

The solution based on perceptron has the advantage of simplicity; the time needed for the structure to learn the weight corresponding to each sampling step value is low, having a negligible overhead. Due to the fact that it has only one layer, the architecture is very sensitive to fast input variations. Thus, this solution is the best choice for the prediction of parameters presenting very fast tendency modifications. Experimental results, presented in Section IV highlight this property.

In contrast, the solution based on Cascade Correlation Neural Network architecture presents a complex structure, which is dynamically created according to the evolution of the inspected parameter. The response to unexpected behavior will be less accurate than the perceptron case because the convergence of this solution is slower, as it needs to modify not only the weights on a single layer, but the weights corresponding to multiple layers. Thus, it perfectly suits for the prediction of parameters having fast variations but no rapid tendency modifications. The 'free memory' scenario presented in Section IV highlights these properties.

We have also to mention that the overhead implied by our approach is comparable to the overhead of any other time series prediction algorithm. Basically, at each prediction step, we will compute the next value for each sampling step we take into account. The number of different time series to consider is a tradeoff between the overhead and the precision of the resulting estimation. As the number of different sampling steps increases, both the overhead and the accuracy also increase. The experimental results have proved that the number of sampling rates equal to three is best choice.

The time complexity for each sampling series is $O(w)$, where w represents the window size. The time complexity for the entire algorithm using the Cascade Correlation approach in the composition stage is $O(w*n) + O(h*n+h^2)$, where:

- n is the number of sampling steps considered
- h is the number of hidden layers dynamically added to the network.

The first term $(O(w*n))$ denotes the complexity of the decomposition stage while the second $(O(h*n+h^2))$shows the overhead of the composition stage.

The experiments have shown that the number of necessary hidden layers is always lower than the number of network inputs, in our case equal to n. Thus, we can suppose that $O(w*n) + O(h*n+h^2) = O(w*n) + O(n^2+n^2) = O(w*n+n^2)$.

In the second case, using a perceptron, the resulting time complexity is $O(w*n) + O(n) = O(w*n)$. Considering that $n = 3$ the overhead implied by our approach is comparable to the overhead of other similar algorithms.

TABLE II
AVERAGE PERCENTAGE ERRORS (%) FOR ONE-STEP AHEAD PREDICTION

Param	SMA	EMA	WMA	Random	CascorGA	FA	FC
cpu idle	0.103	0.116	0.099	0.098	0.019	0.010	**0.009**
cpu usage	6.672	7.284	6.898	8.472	6.433	6.994	**5.645**
load5	0.721	0.832	0.561	0.911	0.481	0.456	**0.266**
free mem	12.107	12.766	9.886	14.266	12.458	8.749	**7.400**

V. EXPERIMENTAL RESULTS

This section presents the experimental results obtained using the new prediction approach. We also make a comparison with the classical prediction algorithms for different types of parameters representing one stations state, information very important and useful in a distributed system. The experiments are made for both one-step ahead and multi-step ahead prediction. The performance criterion used in order to evaluate the results is Absolute Percentage Error (APE), criterion described in Section II.

We have chosen to monitor systems involved in the ALICE [2] project, one of the largest experiments in the world devoted to research in the physics of matter at an infinitely small scale. The system parameters chosen in prediction are discussed in the following paragraph.

A. Predicted Parameters

A prediction based scheduler can improve considerable the time necessary to schedule tasks with dependencies. Using predictions, we can presume that the idle time of tasks can be reduced significantly, resulting a completely resource utilization and load balance. In order to tune a scheduler with a state prediction component, we have to choose the minimum, but the sufficient set of system parameters that is able to provide a correct image over the system state and also a relevant entry for the scheduling component but without a considerable overhead.

In our experiments, we considered the parameters set composed by the cpu usage, the free memory, the swap memory and the load averages. In the following paragraphs we make a comparison between the results of our new approach and other existing prediction algorithms. For the comparison, we use the results obtained for each parameter presented above.

B. Errors Interpretation for One-Step Ahead Prediction

This section discusses the experimental results obtained by our new algorithm in the situation of one-step ahead prediction for the earlier described parameters. The results are also compared with those obtained by other algorithms on the same historical data set, comparison made using the absolute percentage error criterion. The presented graphics highlight the errors obtained by the following prediction approaches: SMA (simple moving average), EMA (exponential moving average), WMA (weighted moving average), Random (random prediction), CascorGA (the prediction algorithm proposed in [12] based on the Cascade Correlation NN and genetic algorithms), FA (the new prediction approach based on composition using a perceptron), FC (the new prediction approach based on composition using the Cascade Correlation NN architecture).

Table II presents the absolute percentage errors obtained using those prediction algorithms for different system parameters where the minimum error for each parameter is bolded. As the table emphasized, in each situation the minimum error is obtained using the new prediction approach using the Cascade Correlation NN architecture.

Figure 8 presents a comparison between the errors obtained for the 'idle' parameter using different prediction algorithms. Because of the fact that the parameter's behavior don't vary very fast, the errors obtained are very small (less than 0.12% for all algorithms). There aren't large differences between the results obtained using the new prediction approach using a perceptron (marked with FA in the graphic) and the results obtained by the new algorithm using a Cascade Correlation NN architecture (marked with FC in the graphic). The difference between those two is 1.8%. The improvement offered by the new approach compared to the algorithm proposed in [12] is more than 51% and more than 90% compared to the classical prediction algorithms [3].

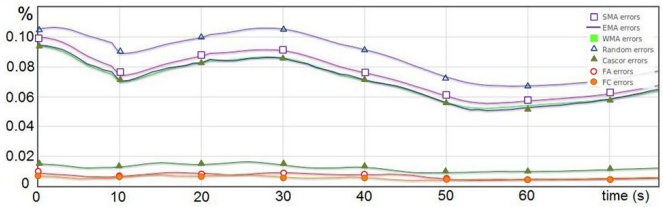

Fig. 8. Errors' evolution for the 'idle' parameter

Figure 9 presents the evolution of the prediction results for the 'load5' parameter using the new approach and the Cascade Correlation NN architecture compared to the real parameter's behavior. Figure 10 present the errors obtained for the 'load5' by different algorithms. The fast variations of the parameter's behavior cause larger errors than the first presented experiment (7% or less). The new prediction approach using the Cascade Correlation NN architecture proved to offer better results than the one using a perceptron, obtaining an error reduced with 30%. In this situation, our new approach offers a great improvement compared to the classical prediction algorithms: more than 38%.

Figure 11 presents the evolution of the prediction results for the 'cpu_usage' parameter using the new approach and

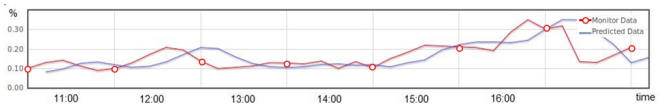

Fig. 9. Prediction' evolution for the 'load5' parameter using the new approach and the Cascade Correlation NN architecture

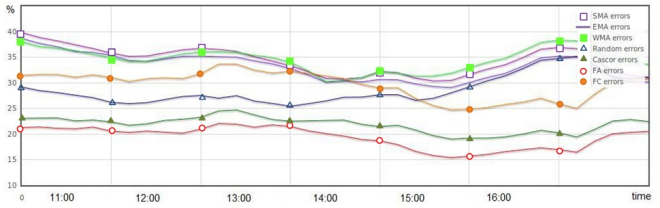

Fig. 10. Errors' evolution for the 'load5' parameter

the Cascade Correlation NN architecture compared to the real parameter's behavior. Figure 12 present the errors obtained by all prediction algorithms. This parameter has fast variations and tendency modifications. The new prediction approach using one perceptron proved to offer better results, obtaining an error reduced with 12.8%.

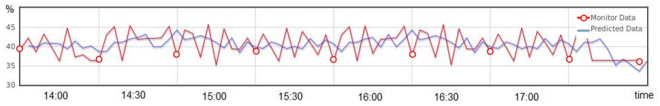

Fig. 11. Prediction' evolution for the 'cpu_usage' parameter using the new approach and the Cascade Correlation NN architecture

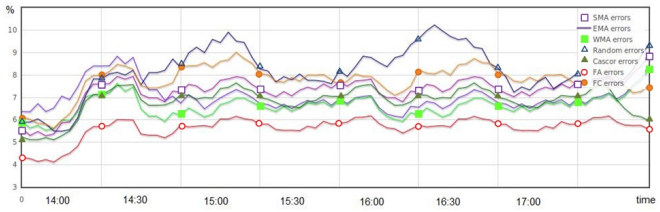

Fig. 12. Errors' evolution for the 'cpu_usage' parameter

A very interesting situation is encountered in the case of 'free mem' parameter prediction, situation represented in Figure 13. This parameter is very hard to predict because it has fast and vast value variations and tendency modifications and this is the reason why the errors are much more significant than those earlier discussed. Analyzing the experimental results, we observe three regions in the graphic, separated in Figure 13 using a vertical line. The first and the third zone are characterized by fast variations but no rapid tendency modifications and our new prediction approach based on cascade correlation architecture proved to offer very good results, improving the error with 15.4%. But in the second zone, characterized by very rapid tendency modifications, this architecture proved to be not proper to predict such a time series, in opposition with the decomposition prediction approach combined with a perceptron who offered the smallest error.

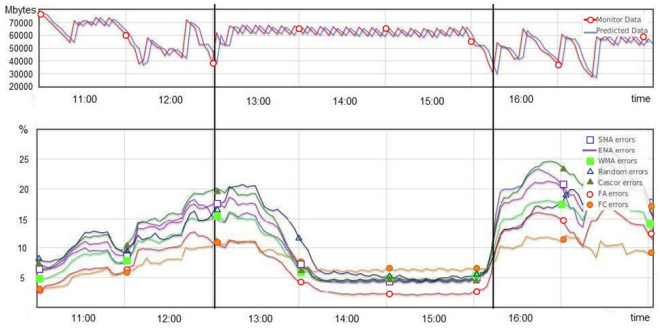

Fig. 13. a) Prediction' evolution for the 'free mem' parameter using the new approach and the Cascade Correlation NN architecture b) Errors' evolution for the 'free mem' parameter

C. Errors Interpretation for Multi-Step Ahead Prediction

A most relevant performance benchmark over all algorithms is given by the situation of multi-step ahead prediction. Also, one-step ahead prediction is often insufficient because, in order to optimize the resources utilization, a meta-scheduler needs an image over the future resources' state on a full period of time not only for a future moment of time.

Short-term prediction is preferred over a long-term one [6], [11] because on the fact that the scheduling mechanism must take decisions on process migration considering limited time and environment constraints. Besides, long-term prediction is not always as beneficial in distributed systems, as another task may start its execution, altering the node behavior unexpectedly. Short-term prediction allows one to schedule the same task over different nodes if its behavior changes when necessary. Summarizing, it is more important to know what may happen in the next few seconds than in several hours.

The multi-step ahead prediction experiments are made for the same system parameters, using the same performance criterion as in the previous case. The tested algorithms were put in the situation to predict the next interval of seven moments of time. The window for all classical algorithms is considered five and all approaches based on neural networks were trained using a training set with 30 elements.

Figure 14 presents the multi-step ahead prediction's results obtained by our approach and classical prediction algorithms for the 'idle' parameter. Figure 15 presents the errors' evolution obtained in this case. As expected, for each algorithm, the errors are increasing more and more for each new prediction step. For all algorithms except FA (our new algorithm using a perceptron in the composition stage) the growth rate is similar for more than 5 future moments. Best results were obtained using our new prediction algorithm that uses in the composition stage a perceptron.

As experimental results emphasize, the multi-step ahead predictions are no very accurate and the errors are most significant than those obtained for the one-step ahead prediction because they are made using previous predictions, meaning that the error in the previous step is amplified in the further step. The accuracy is decreasing with each step of prediction

and at some point it gets to make predictions just based on previous predictions made, meaning an error amplifier.

Best results were obtained by our approach using a perceptron in the composition stage (the orange line in Figure 15). The improvement compared to the solution using in the composition stage consisting the Cascade Correlation architecture is 32%. Compared to the classical predicted algorithms, we obtained a great improvement - 73%.

The prediction algorithms based on mean and standard deviation proved to be unable of accurate multi-step ahead prediction because of several factors. First of all, it should be mentioned their static prediction way (due to their fixed weights, inadequate to the current behavior and tendency of the system parameter's values). Also, the results of a (simple or weighted) mean cannot be greater than the maximum of the time series. If the time series we want to predict has a increasing tendency, such an algorithm will provide poor results because of this limitation. The decreasing situation is similar; the result of a mean algorithm cannot be smaller than its smallest input value.

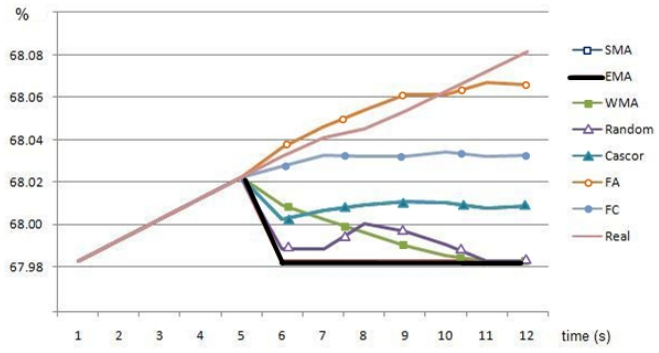

Fig. 14. Multi-step ahead prediction results of the 'idle' parameter

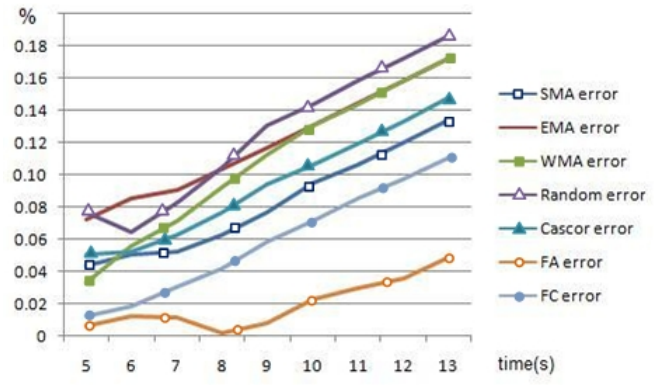

Fig. 15. Multi-step ahead prediction APE errors of the 'idle' parameter

Taking into account those observation, one can conclude that a complex architecture and learning algorithm are needed. Our prediction approach inspired from wave decomposition offered better results and a great error improvement.

VI. Conclusions

We proposed a new prediction algorithm inspired from the decomposition of a complex wave into simpler waves with fixed frequencies. In the decomposition stage, the time series is divided with different sampling steps into multiple time series. For each of these obtained series, we predict the value using one of the classical algorithms. Then, all these results are combined into a single value using methods inspired from artificial intelligence. The experimental results highlighted the great improvement, discussed in terms of error reduction, offered by this approach. The errors were significantly reduced compared to other prediction algorithms and the results are closer to the real system's behavior for tests conducted on real scenarios with data collected form the Alice Experimet.

This paper also presented a monitoring tool needed to provide accurate information about one system state, information very useful in a distributed system for a better understating of its behavior and potential anomalies.

Those monitoring and prediction information can contribute to improve the functionalities and the performance of resource management systems in various ways. The immediate use of our work may be in a prediction based meta-scheduler in order to improve the scheduling decisions assuring the load balancing and optimizing the resources utilization.

VII. Acknowledgments

The research presented in this paper is supported by national project DEPSYS - Models and Techniques for ensuring reliability, safety, availability and security of Large Scale Distributes Systems, Project CNCSIS-IDEI ID: 1710.

References

[1] Sac '08: Proceedings of the 2008 ACM symposium on Applied computing, 2008. Conference Chair-Wainwright, Roger L. and Conference Chair-Haddad, Hisham M.

[2] Catalin C. Cirstoiu, Costin C. Grigoras, Latchezar L. Betev, Alexandru A. Costan, and Iosif Charles Legrand. Monitoring, accounting and automated decision support for the alice experiment based on the monalisa framework. In *GMW '07: Proceedings of the 2007 workshop on Grid monitoring*, pages 39–44, New York, NY, USA, 2007. ACM.

[3] Peter A. Dinda and David R. O'Hallaron. An evaluation of linear models for host load prediction. In *HPDC '99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, page 10, Washington, DC, USA, 1999. IEEE Computer Society.

[4] Evgueni Dodonov and Rodrigo Fernandes de Mello. A novel approach for distributed application scheduling based on prediction of communication events. *Future Generation Computer Systems*, (3):241–250, 2009.

[5] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, pages 524–532, San Francisco, CA, USA, 1990. Morgan Kaufmann Publisher Inc.

[6] A.F.R. Araujo G.A. Barreto. Identification and control of dynamical systems using the self-organizing map. *IEEE TNN on Temporal Coding*, pages 1244–1259, 2004.

[7] Sertan Girgin and Philippe Preux. Basis function construction in reinforcement learning using cascade-correlation learning architecture. In *ICMLA '08: Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, pages 75–82, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Iosif Legrand. Monalisa site. http://monalisa.caltech.edu/monalisa.htm, 2010.

[9] Iosif Legrand, Ramiro Voicu, Catalin Cirstoiu, Cos-tin Gri-go ras, Latchezar Betev, and Alexandru Costan. Monitoring and control of large systems with monalisa. *Queue*, 7(6):40–49, 2009.

[10] Florin Pop, Alexandru Costan, Ciprian Dobre, and Valentin Cristea. Prediction based meta-scheduling for grid environments. In *CSCS-17, 17th International Conference on Control Systems and Computer Science*, pages 128–136, Bucharest, Romania, 2009.

[11] Mahdi Aliyari Shoorehdeli, Mohammad Teshnehlab, Ali Khaki Sedigh, and M. Ahmadieh Khanesar. Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods. *Appl. Soft Comput.*, 9(2):833–850, 2009.

[12] Andreea Visan, Mihai Istin, Florin Pop, and Valentin Cristea. Automatic control of distributed systems based on state prediction methods. In *ADiS - First International Workshop on Autonomic Distributed Systems*, Cracow, Poland, 2010.