Grid Global Behavior Prediction

Jesús Montes
CeSViMa
Universidad Politécnica de Madrid
Madrid, Spain
Email: jmontes@cesvima.upm.es

Alberto Sánchez
E.T.S. de Ingeniería Informática
Universidad Rey Juan Carlos
Madrid, Spain
Email: alberto.sanchez@urjc.es

María S. Pérez
Facultad de Informática
Universidad Politécnica de Madrid
Madrid, Spain
Email: mperez@fi.upm.es

***Abstract*—Complexity has always been one of the most important issues in distributed computing. From the first clusters to grid and now cloud computing, dealing correctly and efficiently with system complexity is the key to taking technology a step further. In this sense, global behavior modeling is an innovative methodology aimed at understanding the grid behavior. The main objective of this methodology is to synthesize the grid's vast, heterogeneous nature into a simple but powerful behavior model, represented in the form of a single, abstract entity, with a global state. Global behavior modeling has proved to be very useful in effectively managing grid complexity but, in many cases, deeper knowledge is needed. It generates a descriptive model that could be greatly improved if extended not only to explain behavior, but also to predict it. In this paper we present a prediction methodology whose objective is to define the techniques needed to create global behavior prediction models for grid systems. This global behavior prediction can benefit grid management, specially in areas such as fault tolerance or job scheduling. The paper presents experimental results obtained in real scenarios in order to validate this approach.**

I. Introduction

Large scale distributed systems have paved the way to face complex, technical and scientific challenges that can not be solved with traditional systems, due to their enormous computing and/or storage requirements. Initiatives such as BOINC [1], PlanetLab [2] or TeraGrid [3] and, more generally speaking, grid [4] or the recent cloud computing [5] provide computing and storage resources that can be scaled to a level difficult to imagine elsewhere. Nevertheless, the complexity of these environments makes their management difficult.

System understanding is the key to managing these systems. For this purpose, a deep knowledge about the behavior of each single element is usually required. However, the extraordinary number of different resources makes it almost impossible to analyze and assign efficient policies to every one. Most current grid management techniques are based on this approach [6]–[8], dealing with each independent resource's behavior separately. A good alternative is to simplify the understanding of the system as a whole, studying it as a single entity instead of the set of elements that together constitute it.

Following this idea, we rely on a methodology to model the global behavior of large-scale distributed systems [9], [10] (from now on it will be named Global Behavior Modeling (GloBeM)). GloBeM's aim is to identify regularities in global grid behavior that can be explained. On the other hand, the knowledge of not only the current system behavior, but also of future behavior makes it possible to improve system management. Nevertheless, predicting system behavior is one of the most challenging tasks due to the complexity and heterogeneity of a grid. In this paper, our approach combines the use of machine learning prediction techniques with a *single entity* vision of the grid in order to improve the management of the whole system. Related research has focused on resource-related management, while our approach uses this *single entity* vision to focus on service-related global aspects.

The paper is organized as follows. Section II presents the GloBeM methodology to model a very complex environment as a single entity. Section III shows the basis for this work and an *a-priori* study to obtain an initial framework. Section IV proposes two ways (simple and advanced) to build a system prediction based on global behavior modeling. Section V shows the evaluation of the proposal comparing it with the other possibilities presented. Section VI describes previous works related to the problem herein described. Section VII presents the main conclusions and outlines future work.

II. GloBeM

GloBeM is a methodology for modeling the global behavior of a grid [9], [10]. Its main objective is to build an abstract, descriptive model of the global system state. This enables the model to implicitly describe the interactions between entities, which has the potential to unveil non-trivial dependencies significant for the description of the behavior, which otherwise would have gone unnoticed. This unique features make GloBeM particularly useful in grid management, especially because they provide the means to capture complex interactions among components in a simple yet comprehensive finite state machine behavior model whose states can be directly mapped to the behavior patterns we want to identify.

GloBeM methodology aims at constructing models with the following four general characteristics:

- **Specific state definition:** State characteristics and transition conditions are unambiguously specified. The number of states is minimal for usability reasons.
- **Stability:** The resulting model is a close approximation of the behavior of the system in time, both with respect to the environment and the usage scenario of the service.
- **Simplicity:** The resulting model is easy to understand and provides meaningful insight into the system's behavior.
- **Relevance to service:** The model states are semantically related to the functionality provided the system. This ensures that the observed behavior can be explained in terms of what is expected from the service, thus enabling correlations to quality of service.

GloBeM follows a set of procedures in order to build such a model, starting from monitoring information that corresponds to the observed behavior. These basic monitoring data are then aggregated into *global monitoring parameters*, representative of the global grid behavior instead of each grid resource separately. This aggregation can be performed in different ways, but it normally consists in calculating global statistic descriptors (mean, standard deviation, skewness, kurtosis, etc.) values of each basic monitoring parameter for all grid resources present. This ensures that global monitoring metrics are still understandable from a human perspective. This global information undergoes a complex analysis process in order to produce a global behavior representation. This process is strongly based on machine learning and other knowledge discovery techniques, such as virtual representation of information systems (VR spaces) [11], [12]. Figure 1 depicts this process.

A behavior model presents the following features:

- **Finite state machine**: The model can be expressed as a finite state machine, with specific states and transitions. The number of states is usually small (between 3 and 8).
- **State characterization based on monitoring parameters**: The different system states are expressed in terms of the original global monitoring parameters. This ensures that its characteristics can be understood and directly used for management purposes.
- **Extended statistical information**: The model is completed with additional statistic metrics, further expanding the state characterization.

III. Initial considerations and a-priori study

GloBeM models provide useful information about the system behavior, but they are strictly descriptive in nature. They can be used to understand and optimize a grid, but they provide little knowledge about the system evolution

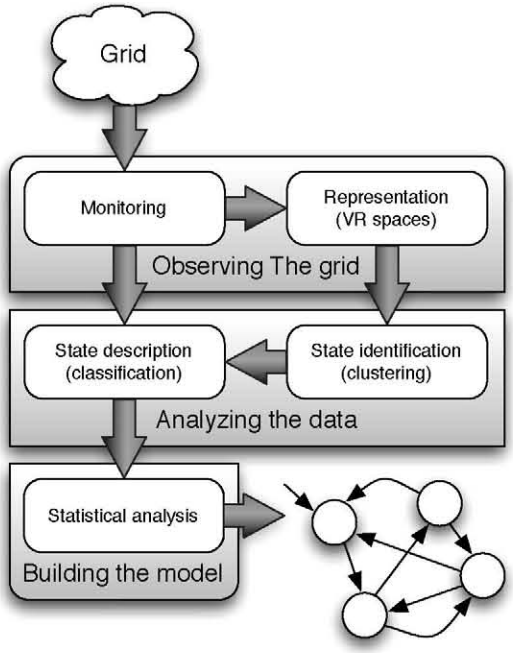

Figure 1. How GloBeM works

over time and/or its future state. In order to further increase their usefulness, GloBeM models could be combined with predictive techniques, capable of foreseeing future events. This would enable the management system to act before such events actually occur, avoiding global faults or any other possibly dangerous situation and improving performance and/or dependability.

In this paper we present a set of algorithms designed to create global state prediction models in terms of GloBeM behavior descriptions. They are based on machine learning and time series analysis techniques. A set of basic elements can be distinguished in all of them:

- The **set of grid states** $S = \{s_1, s_2 ... s_n\}$.
- The **behavior model** $B(t)$ generated using the GloBeM methodology. It describes the grid states S and the events that cause a transition from one state to another. At any instant t, $B(t) = s_k \mid s_k \in S$, where s_k is the grid state in that instant.
- The **prediction model** $P(t)$ that predicts the futures states indicated by the behavior model. At any instant t, $P(t) = B(t+1)$.
- The **training data**. This is the historical grid monitoring data set used to create the behavior and prediction models. It contains a log of values of the monitoring parameters used by the behavior model in order to determine the current state and the associated global state.

- The **test data**. This is a different set of historical grid monitoring data. Although it is similar to the training data, it is much larger and it is used to evaluate the prediction model accuracy.

In basic terms, given a set of current monitoring values, the behavior model indicates the current grid state, but it provides no information about the future. Given the same set of values and a history of past ones the prediction model will be able to predict the future state. The accuracy of this prediction will depend on the quality of the training data and the algorithm used to generate the prediction model. To measure this accuracy, we use the $F1$ score [13]. This is a statistical measure of a test accuracy that can be interpreted as a weighted average of the *precision* and *recall* of a certain classification. An $F1$ score reaches its best value at 1 and worst score at 0. We consider our prediction model as a classifier (it classifies each instant as belonging to the class associate with the future state) and we test it through the test data. *Precision* is defined as the number of correctly predicted instants by the model divided by the total number of predicted instants and *recall* is defined as the number of correctly predicted instants divided by the total number of existing instants. In a more formal way, *precision* and *recall* can be defined with the following equations:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (1)$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (2)$$

Finally, the $F1$ score is defined as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

As it can be seen, the $F1$ score measures the predictor's accuracy in a better way a simple percentage of correctly classified instances would do, since it incorporates information about false positives and false negatives as well as true positives. This is an accurate statistical measure, widely used and commonly accepted in the scientific community.

The test data is used to calculate the two different versions of the $F1$ score for any given prediction model $P(t)$:

- Total $F1$: Score calculated for the whole test data set.
- Transitions $F1$: Score calculated only for the state transitions observed in the test data set. This score helps to evaluate how well state transitions are predicted.

The second value is important because it has been demonstrated that GloBeM's models of these environments tend to be very stable [10] (GloBeM hides the vast complexity of a large distributed system) and state transitions represent major changes in the system behavior. From a general perspective, predicting global state transitions could be the key to improving grid management in many areas (job scheduling, dependability and fault tolerance, etc). In most cases a change of global state will require a change in the management policies, specially to prevent undesirable situations or states where some service requirements are not met (faults and/or failures, decreases in quality of service, etc). A prediction model strongly benefits the management system in these critical situations (transitions), making it possible to anticipate and act ahead of faults and changes. The best prediction models would score highly on total and transitions $F1$ scores.

A. A-priori study

In order to obtain a basic framework, an *a-priori* study was made. For this study real monitoring data from PlanetLab [2] were used. PlanetLab is a global scientific research network, used by researchers at top academic institutions and industrial research labs to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. PlanetLab currently[1] consists of 1138 nodes at 519 sites, scattered all over the world. It presents all the heterogeneity, complexity and variability expected from any real grid computing infrastructure, and therefore it is an excellent scenario for testing the global behavior prediction techniques presented here.

PlanetLab provides free access to a monitoring tool called CoMon [14], which is capable of presenting detailed information about the current state of each active node in the system. Many different parameters are monitored, including CPU usage, memory usage, network traffic, architecture characteristics, I/O operations, an so on. Information from this tool is being gathered in order to create a comprehensive monitoring database of the historical evolution of PlanetLab. For this study a total of 8 months of PlanetLab monitoring information were used. Data was aggregated in 1 hour monitoring intervals and divided in many subsets, in order to produce an extensive collection of training sets. Subsets sizes ranged from 10 to 110 days, with different degrees of overlapping between them. Altogether a set of 220 training subsets was created.

Using this training data set, different behavior models were produced in order to explain the behavior observed in the whole data set. Then, for each behavior model, statistics about percentage of transitions and state stability were calculated.

Table I
A-PRIORI STUDY RESULTS

	mean	standard deviation
Stable periods	90.1%	2.51
State transitions	9.9%	2.51
Stable period duration	18.28h	8.22

[1]November 2010

Summary values of the *a-priori* study are presented in Table I. As can be seen, the average number of state transitions is quite low ($\simeq$ 10%), which illustrates the previously stated idea that GloBeM models are very stable, with few but relevant transitions. These transitions, however, are crucial events, representing major changes in the system behavior that normally involve clear modifications in aspects such as performance or dependability. From a management point of view, these are the key situations that need to be anticipated, creating the need of a prediction model capable of foreseeing state transitions.

As part of the *a-priori* study, a basic predictor was constructed, in order to provide a basis for evaluation and comparison. This was called the *naïve* predictor.

The naïve predictor

As the simple prediction model reference for the *a-priori* study, the *naïve* predictor $P_N(t)$ was defined in the following terms:

$$P_N(t) = B(t) \tag{4}$$

This basically means that $P_N(t)$ will always predict the future state to be the current state, as given by the behavior model. In consequence, the prediction will be correct as long as no state transition occurs. When the transition takes place, the $P_N(t)$ predictor fails, as it always expects the state to remain stable. The accuracy of this *predictor-that-does-not-predict* will obviously depend on the stability of the system, as it only fails when transitions occur. A study of the $F1$ score values would provide a basic frame of reference for prediction models evaluation, defining when predicting is actually better than a simple descriptive approach with no anticipation.

Using the 220 PlanetLab monitoring training sets to generate different behavior models, the accuracy of the *naïve* predictor was evaluated. Table II shows the predictor accuracy metrics for $P_N(t)$. As it can be seen in table II, Even though the *naïve* predictor is incapable of predicting any transitions ($F1$ (transitions) $= 0.0$), the total average is very high ($F1$ (total) $= 0.87$). This is consistent with the statistical results presented in Table I. The system is very stable with very few state transitions. Nevertheless, detecting these transitions is our main objective, as these are the relevant events that are identified and give meaning to the GloBeM behavior model.

Table II
NAÏVE PREDICTOR ACCURACY METRICS

$F1$ (total)	0.87
$F1$ (transitions)	0.0

IV. PREDICTING GLOBAL BEHAVIOR

As was explained, state transitions in GloBeM behavior models indicate crucial events in the system, usually requiring the adaptation of global management policies. In this section we present two approaches to global state prediction, in order to anticipate future states and state transitions in a grid system. The first one is a basic, single variable prediction strategy, based on traditional time series analysis techniques and machine learning. The second one is a far more complex, multi-stage approach, introducing some advanced concepts.

A. Basic predictor

Considering the system's global state as a variable, at a given time t $B(t) = s_t \mid s_t \in S$, and therefore s_{t-1} would be the state at time $t-1$, s_{t-2} the state at time $t-2$, and so on. We can consider the associated time series as:

$$S_t = \{s_t, s_{t-1}, s_{t-2}, s_{t-3}, ...\}$$

For any given instant in time t, S_t will contain the past and present state values of the system, showing its historical evolution.

Using traditional time series analysis techniques, we define our basic predictor model as a function capable of calculating the future state based on the present and past values of the global state time series variable S_t:

$$P_B(t) = f(s_t, s_{t-1}, s_{t-2}, s_{t-3}, ...) \tag{5}$$

In practical terms, there is only so many instants in the past that can be considered and therefore we redefine $P_B(t)$ as:

$$P_B(t, w) = f(s_t, s_{t-1}, s_{t-2}, ..., s_{t-w}) \tag{6}$$

where w is the number of past values considered in the prediction. We call w the **predictor window**.

The $P_B(t, w)$ algorithm consists of three phases, aimed at creating a prediction model for a GloBeM behavior model. These phases are illustrated in Fig. 2 and described below:

1) **Training data classification**: Using the behavior model, the training data are classified, in order to determine the state associated to each monitoring instant. The result is an extended version of the training data set, including the state variable along with monitoring parameters.
2) **Time series selection**: The values from the state variable in the training data set are selected, generating the S_t time series.
3) **Machine learning**: Using a machine learning algorithm, a prediction model is trained using data from the S_t time series. The number of past values the machine learning algorithm can include in its calculations is determined by the w value defined above.

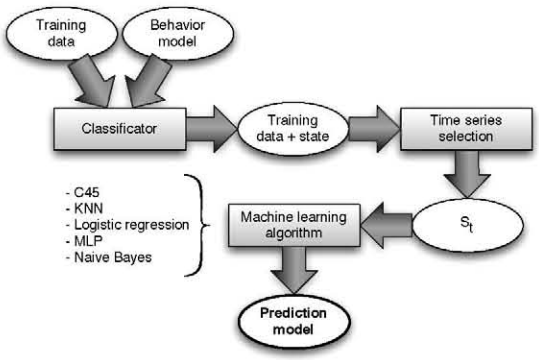

Figure 2. Basic predictor phases

The result is a prediction model $P_B(t)$ for the S_t time series. The exact form of this model depends on the machine learning algorithm used. At this point, instead of selecting one specific algorithm, we have proposed the following set of them:

- C4.5 [15] is a statistical classifier of the ID3 family of algorithms that generates a model in the form of a classification tree. The leaf nodes of the decision tree contain the class name, whereas a non-leaf node is a decision node. The decision nodes represents attribute tests, with each branch (to another decision tree) being a possible value of the attribute. The C4.5 algorithm extends ID3 providing mechanisms to deal with continuous and missing values.
- The K-Nearest Neighbors algorithm (KNN) [16] is a classifier algorithm based on agreement. Thus an object is assigned to the class most common amongst its k closest neighbors.
- Logistic regression [17]. The aim of a regression analysis [18] is to know the statistical relation existing between a dependent variable and one or more independent variables. In this sense, a functional relation between the variables must be postulated. In this case, data are fit to a logistic curve.
- A Multi-Layer Perceptron (MLP) [19] is an artificial neural network model that selects the corresponding output for the specific input data. The MLP extends the standard linear perceptron using several layers of neurons.
- Naïve Bayes [20], [21] is based on applying Bayes' theorem. This classifier is a model of conditional independence of predictor attributes, ensuring an optimal classification if explicit assumptions are met.

Our objective is to provide an extensive set of machine learning algorithms, in order to present as complete a study as possible. The five selected techniques are well known, widely used and scientifically relevant. In Section V, experimental results are presented to illustrate which machine learning technique is more adequate in our case, and the overall performance of the basic predictor.

B. Multi-stage predictor

After the basic predictor $P_B(t)$ was developed, the need of a more advanced prediction technique appeared, motivated by several issues.

First, as shown in the *a-priori* study, the amount of state transitions observed in the GloBeM models is quite low. Training data sets are composed mostly of data that represent stable instants where no transition takes place. When this training data sets are used in machine learning algorithms, they usually lead to prediction models that are over-fitted to predict stability and less concerned with transitions. In situations where the disproportion among stable instants and transitions is extreme, the machine learning algorithm basically disregards transitions, as they represent a very uncommon situation.

Second, the behavior state variable (and its associated time series S_t) is clearly dependent on the global monitoring parameters, as it is derived from them by the GloBeM model. This information is not included in the $P_B(t)$ model, which limits its efficiency. In order to deal with these issues, a more complex predictor was developed. We consider again the state time series S_t and the predictor window w. We incorporate also the set of global monitoring variables $\{V_1, V_2, ..., V_M\}$ **selected by GloBeM**[2] to construct the behavior model and the associated time series for each one:

$$\begin{aligned} V1_t &= \{v1_t, v1_{t-1}, v1_{t-2}, ...\} \\ V2_t &= \{v2_t, v2_{t-1}, v2_{t-2}, ...\} \\ &... \\ VM_t &= \{vm_t, vm_{t-1}, vm_{t-2}, ...\} \end{aligned}$$

As it is stated below, these values are not basic resource monitoring metrics. GloBeM uses global, aggregated parameters to generate its behavior model. These parameters are calculated using basic monitoring metrics, and then automatically selected, identifying those that carry the most relevant information. The time series $V1_t, ..., VM_t$ are generated using those global aggregated monitoring parameters selected by GloBeM as representative of the grid behavior. We define the predictor $P_M(t)$ as follows:

$$P_M(t) = f(s_t, ...s_{t-w}, v1_t, ...v1_{t-w}, ...vm_t, ...vm_{t-w}) \quad (7)$$

As can be seen, the first difference between $P_M(t)$ and the previous $P_B(t)$ is that the global monitoring parameters are also considered in the prediction, and not just the present and past state. In addition, $P_M(t)$ improves the transition prediction accuracy by means of a multi-stage prediction

[2]GloBeM selects automatically the variables that has a high influence on the model description

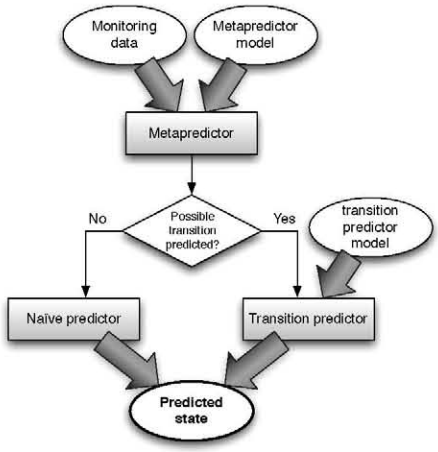

Figure 3. Multi-stage predictor

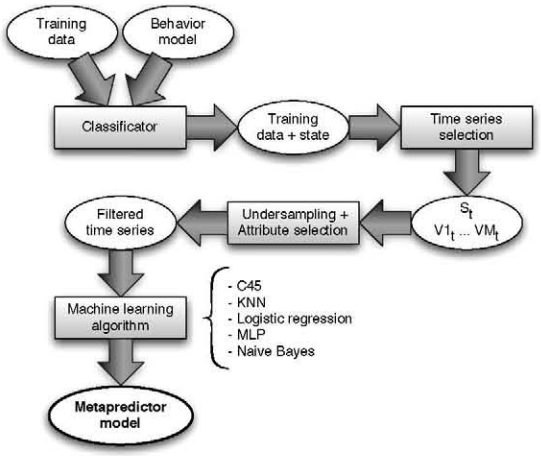

Figure 4. Metapredictor model construction phases

process. This process structure can be seen in Fig. 3. The multi-stage predictor is composed of three basic elements:

- The **metapredictor** $MP(t)$ is a prediction model trained to predict state transitions. It is capable of foreseeing whether the system is going to change state, but not the specific state it is going to transit to.
- The ***naïve*** **predictor** $P_N(t)$, as defined in Section III-A, is used when the metapredictor indicates that no transition is going to happen. This strongly simplifies the prediction process in those cases, as no prediction is really made.
- The **transition predictor** $P_T(t)$ is a prediction model trained to anticipate only state transitions. It is trained using global monitoring data from instants in time when state transitions happen, and therefore it is generated specifically for those situations. The transition predictor is used when the metapredictor anticipates a transition, maximizing the probability of correct prediction in those cases without affecting the general prediction accuracy.

As can be seen in Fig. 3, the multi-stage predictor uses its metapredictor to determine if transitions are going to happen. In case a global state transition is anticipated, the multi-stage predictor then relies on its transition predictor to determine the future state. In case no transition is foreseen, the multi-stage predictor simply anticipates no change, providing the *naïve* predictor result as its final prediction.

The metapredictor

The construction of the metapredictor model is carried out in four phases, as shown in Fig. 4. The process is similar in essence to the one previously described for the basic predictor, but more complex. The four metapredictor model construction phases are:

1) **Training data classification**: In the same way as for the $P_B(t)$ model, the training data are classified using the behavior model, in order to determine the state associated to each monitoring instant.
2) **Time series selection**: In this case not only the values from the state variable are selected, but also the ones from the global monitoring variables, creating the time series set $\{S_t, V1_t, ..., VM_t\}$.
3) **Undersampling and attribute selection**: In order to increase the quality of the time series training set generated, two special refining techniques are used in this phase. These are *undersampling* and *attribute selection* and they are described in detail below.
4) **Machine learning**: Finally the machine learning algorithm is executed in this phase, in a similar way as in the basic predictor. As in that previous case, we selected five possible algorithms to be used: C4.5, KNN, logistic regression, MLP and Naïve Bayes.

Undersampling is a data filtering technique commonly used in machine learning procedures where, given a classification of a training set, the proportions in which each class appears are clearly uneven. In our case, if we divide the data set in *stable instants* and *state transition instants*, we find out most of them belong to the first group. As was explained before, in these cases machine learning algorithms tend to focus only on the majority class (stable instants, in our case), almost completely ignoring the minorities. To avoid this phenomenon, the majority class is reduced to a statistically significant subset of values, representative of the whole group but of a size similar to the minority groups (or at least not so overwhelmingly larger). This gives the machine learning technique a chance to correctly identify all classes.

To achieve this we used the k-means clustering algorithm [22]. K-means classifies the data in a specified number of

classes, with similar observations assigned to the same class. As a result, it produces a list of representative values, called centroids, one for each class. To undersample the metapredictor training data set, the observations that represent stable instants (much more frequent than the ones representing state transition instants) are separated and then classified using k-means. The metapredictor algorithm sets a number of classes for the k-means algorithm to the number of state transitions observed, and takes the resulting centroids as representative observations.

A second training set optimization carried out in the metapredictor construction algorithm is attribute selection. As shown in (7), the $P_M(t)$ model is defined from a function of many parameters, basically the present and past values of the system's global state and global monitoring parameters, given a certain predictor window w. When the number of global monitoring parameters and w is high, this will originate a function with a very large set of parameters. Not all these parameters are statistically relevant for prediction purposes. However they increase the training data set size, making the subsequent machine learning process difficult. In order to select only the statistically representative parameters for the machine learning process, the metapredictor algorithm calculates the autocorrelation coefficients for each input time series.

Autocorrelation coefficients [23] are a commonly used time series analysis tool. They indicate the correlation (usually the Pearson correlation coefficient) between present and past values of a time series, at any given time. For instance, a time series of a variable whose value at any time is dependent only on its last two values will score closer to 1 in its two first autocorrelation coefficients and close to 0 in the rest. Calculating this coefficients will indicate that no other past observations are needed in order to predict the variable value. In the metapredictor construction, the first w autocorrelation coefficients are calculated for each time series used ($\{S_t, V1_t, ..., VM_t\}$). Then, only the relevant historical values of each series (score closer to 1) are selected, effectively reducing the number of parameters provided to the machine learning algorithm.

Finally, the machine learning algorithm is configured to generate a model that only predicts whether the system global state is going to remain stable, or a transition will occur. The final model produced is called the **metapredictor model**.

The transition predictor

The second part of the multi-stage predictor is the transition predictor $P_T(t)$. The construction of this prediction model takes place in the following six phases (Fig. 5):

1) **Training data classification**: In the same way as for the $P_B(t)$ and $MP(t)$ models, the training data are classified using the behavior model.

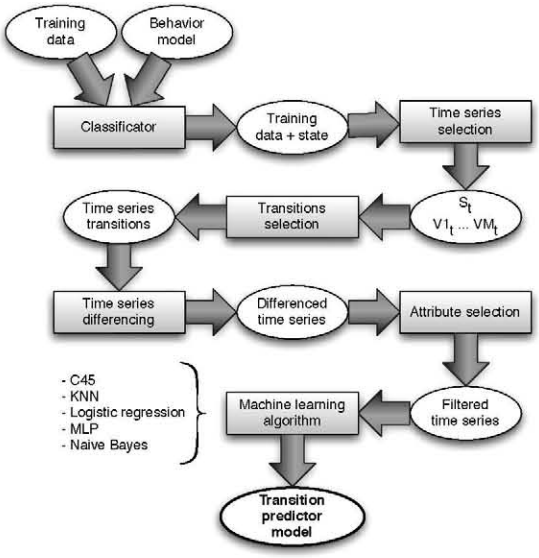

Figure 5. Transition predictor model construction phases

2) **Time series selection**: Like in the case of $MP(t)$, the state and global monitoring variables are selected, creating the time series set $\{S_t, V1_t, ..., VM_t\}$.
3) **Transition selection**: At this point, the time series training set is filtered, in order to select only values related to state transitions. This creates a data set containing only specific information about global changes of state.
4) **Time series differencing**: The time series are differenced in order to remove unnecessary information that could affect the subsequent machine learning process. This process is explained in detail below.
5) **Attribute selection**: In a similar way as in the case of $MP(t)$, time series data set attributes are selected using autocorrelation coefficients.
6) **Machine learning**: Finally the machine learning algorithm is executed in this phase, in a similar way as in the previous predictors. Again we selected the same five possible algorithms to be used: C4.5, KNN, logistic regression, MLP and Naïve Bayes.

Differencing is a commonly used time series analysis tool. Its objective is to eliminate any possible trend in the series, leaving only relevant information about changes in the variable. From a general perspective, if we consider the time series $X_t = \{x_t, x_{t-1}, ...\}$, first order differencing X_t consists in replacing it with a new series Y_t defined as:

$$Y_t = \{y_t, y_{t-1}, ...\}$$
$$\forall y_k \in Y_t, y_k = \nabla x_k = x_k - x_{k-1}$$

During construction of the transition predictor, first order differencing is applied to all numeric series in the time series

training data set in order to provide only useful information to the machine learning algorithm.

Once the six previously explained phases take place, the result obtained is a predictor model specifically trained to detect global state transitions. When it is incorporated inside the multi-stage predictor, it is only used when the metapredictor model indicates a transition will occur. The combined use of $P_N(t)$, $MP(t)$ and $P_T(t)$ carried out by the multi-stage predictor generates a more efficient prediction model than $P_B(t)$, specially anticipating global state transitions. In the following section an experimental study is presented.

V. Experimental results and evaluation

Using the same PlanetLab scenario described in the *a-priori* study (see Section III-A), a series of experimental tests were performed. The objective of these tests was to evaluate the accuracy of the different prediction algorithms proposed, using the previously described metric $F1$. The general characteristics of the test series can be seen on Table III.

Table III
Experiment characteristics

Total size of test data	8 months
Data time resolution	1 hour
Size of training data	60, 70, 80, 90 or 100 days
Total number of training models	100 (20 of each size)
Predictor window (w)	10, 20, 30, 40 or 50 hours
Total number of configurations	500

As presented on the table, several different predictor window values were used. Also training sets of different sizes were included in the experiment series. Each experiment was generated using a specific training set with a fixed w value, giving a total number of 500 experimental configurations. Each experiment was performed using the five machine learning algorithms considered: C4.5, KNN, Logistic regression, MLP and Naïve Bayes. The specific parameters of these algorithms were set to generic purpose values, in order to provide a more general perspective. Table IV shows these configurations. Please refer to the above mentioned references concerning these machine learning algorithms for further analysis of these parameters.

A. Basic predictor evaluation

The basic predictor $P_B(t)$ was evaluated using the experiment series previously described. For each experiment the $F1$ values were calculated. Fig. 6 shows the average results obtained, separated by machine learning algorithm used. The issues previously anticipated in Section IV-B can be clearly seen here, causing a reduction in the predictor accuracy. The total $F1$ value ranges between 0.83 and 0.90 and the transitions $F1$ between 0.13 and 0.22, which is an improvement over the Naïve predictor. However, even though the predictor is capable of anticipating a few transitions, the transition accuracy is too low to be considered acceptable. These results justify the need for a more complex approach.

Table IV
Machine learning algorithms configuration

C4.5	
Confidence factor	0.25
Min. num. objects per leaf	2
KNN	
K	1
Logistic regression	
Log-likelihood ridge	10^{-8}
MLP	
Learning rate	0.3
Momentum rate	0.2
Number of epochs	500
Number of hidden layers	$(attribs + classes)/2$

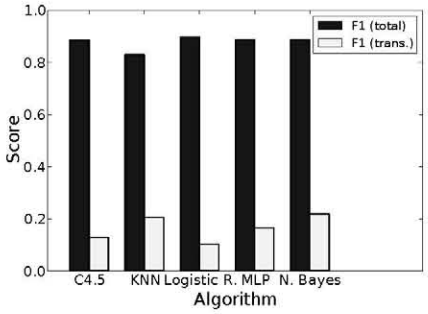

Figure 6. Basic predictor results

B. Multi-stage predictor evaluation

In a similar fashion to the basic predictor, the multi-stage predictor was evaluated, using the five suggested machine learning algorithms. The results differ depending on the machine algorithm used (Fig. 7). All C4.5, KNN and Logistic Regression experiments produced very good total $F1$ scores (between 0.85 and 0.89) which guarantees their overall accuracy. Furthermore, KNN experiments shown also a high transitions $F1$ score (0.63), demonstrating that the multi-stage predictor is capable of correctly anticipating global state transitions. These are considered to be fairly good results, given the intrinsic difficulty of anticipating this rare but critical events. Other algorithms, however, obtain worse results (specially Naïve Bayes). Classification algorithms performance always depends greatly on the characteristics of the data to be classified, and there is no single classifier that produces optimal results for any given problem (a phenomenon that may be explained by the "no free lunch" theorem[3] [24]). Specific reasons behind the differences observed in the multi-stage predictor accuracy when using different machine learning methods would require a study of those algorithms, which is out of the scope of this paper.

[3]The "no free lunch" theorem states that "any two learning algorithms are equivalent when their performance is averaged across all possible problems".

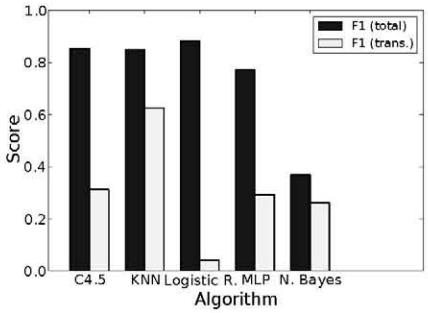

Figure 7. Multi-stage predictor results

VI. Related Work

There are different research works which have addressed the topic of behavior prediction in grid environments.

Rood and Lewis [25], [26] propose a multi-state model as well as several prediction techniques in order to forecast the availability of resources and the transitions into the model's states. This approach can improve scheduler efficiency. Although the goal of this work is similar to ours, the main difference is that in the former case, analysis and prediction are performed at resource level. Our approach simplifies the analysis dealing with a generic model of the grid, making both the analysis and further decision-making approaches easier.

In the work presented by Pietrobon and Orlando [27], the prediction regarding whether a job fails or not is made by means of regressive analysis applied to job running logs. Unlike this work, our approach makes a behavior model of the grid, which can improve scheduler effectiveness.

Li et al. [28] present an Instance Based Learning technique to forecast response times of jobs in grids by means of historical performance data mining. This approach is based on the definition of similarity between jobs. In a similar way, Smith et al. [29] predict the run times of parallel applications from past executions of similar applications. Cho et al. [30] describe a user demand prediction approach, which uses historical user demands in order to manage efficiently grid resources. All these works are focused on user jobs or user demands. Our approach predicts the state of a grid infrastructure, which enables the application of enhanced scheduling techniques that affect to the whole grid.

VII. Conclusions and Future Work

Grid environments are suitable in high demanding scenarios where other computing solutions have traditionally failed. However, one of the weakest aspects of these systems is that they are difficult to manage due to their complexity and dynamism. A good approach to simplifying grid understanding and management is to treat it as a single entity instead of as a set of different elements that together form it [10]. Although this alternative simplifies management, a prediction phase was still required to largely improve it by anticipating crucial changes in system behavior.

As explained in Section III, the process of predicting these crucial changes is not an easy task, since grid systems behave in a very stable way (from a global modeling perspective). Given the stability of behavior models, transitions or behavior changes rarely occur and, therefore, are difficult to predict. This makes any basic statistical predictor incapable of finding such changes. Given a correct selection of the machine learning algorithm, our multi-stage predictor proposal is capable of predicting a high percentage of these transitions, as well as being able to recognize the system stability, as described in Section V. Consequently, the prediction proposed in this paper can significantly benefit grid management systems, enabling one to act ahead of system changes and to select suitable management policies to deal with those changes before they occur.

Regarding future work, we are planning to test the benefits and improvements that our proposal entails in different grid management fields, such as dependability, quality of service, data management and job scheduling. At the same time, we will continue improving our multi-stage predictor scheme with the aim of being able to provide even better accuracy results.

Acknowledgment

This work has been partially funded by the the Spanish Ministry of Education and Science (grants TIN2007-67188 and TSI2007-65677), the *Centro de Supercomputación y Visualización de Madrid* (CeSViMa) and the Madrid Regional Authority (Comunidad de Madrid), under contract CCG10-URJC/TIC-5185. Monitored grid resources used are part of PlanetLab (*http://www.planet-lab.org/*).

References

[1] "BOINC, accessed Mar 2010." [Online]. Available: http://boinc.berkeley.edu/

[2] "PlanetLab: An open platform for developing, deploying, and accessing planetary scale-services, accessed May 2010." [Online]. Available: http://www.planet-lab.org/

[3] "TeraGrid, accessed May 2010." [Online]. Available: http://www.teragrid.org/

[4] I. Foster, "What is the Grid? A Three Point Checklist," *Grid Today*, vol. 1, no. 6, Jul 2002. [Online]. Available: http://www.gridtoday.com/02/0722/100136.html

[5] "Twenty-One Experts Define Cloud Computing, accessed May 2010." [Online]. Available: http://cloudcomputing.sys-con.com/node/612375/print

[6] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid," in *Proc. of the 4th International Conference on High Performance Computing in the Asia-Pacific Region*, vol. 1, China, 2000, pp. 283–289.

[7] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Software Practice and Experience*, vol. 32, no. 2, pp. 135–164, 2002.

[8] M. Siddiqui and T. Fahringer, "GridARM: Askalon's Grid Resource Management System," in *Advances in Grid Computing (EGC 2005) - Revised Selected Papers*, ser. L. Notes in Computer Science, vol. 3470. Amsterdam, Netherlands: Springer Verlag, June 2005, pp. 122–131.

[9] J. Montes, A. Sánchez, J. J. Valdés, M. S. Pérez, and P. Herrero, "The grid as a single entity: Towards a behavior model of the whole grid," in *OTM Conferences (1)*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 5331. Springer, 2008, pp. 886–897.

[10] ——, "Finding order in chaos: a behavior model of the whole grid," *Concurrency and Comp.: Practice and Experience*, p. In press., 2010.

[11] J. J. Valdés, "Similarity-based heterogeneous neurons in the context of general observational models," *Neural Network World*, vol. 12, pp. 499–508, 2002.

[12] ——, "Virtual reality representation of information systems and decision rules," *L. Notes in Artificial Intelligence*, vol. 2639, pp. 615–618, 2003.

[13] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed. London: Butterworths, 1979.

[14] "CoMon - A Monitoring Infrastructure for PlanetLab, accessed May 2010." [Online]. Available: http://comon.cs.princeton.edu/

[15] J. Quinlan, *C4.5: Programs for machine learning*. Los Altos, CA: Morgan Kaufmann, 1993.

[16] B. V. Dasarathy, *Nearest neighbor (NN) norms : NN pattern classification techniques*. IEEE Computer Society Press, 1991.

[17] D. W. Hosmer and S. Lemeshow, *Applied logistic regression*. Wiley Interscience, October 2000.

[18] D. V. Lindley, "Regression and correlation analysis," *New Palgrave: A Dictionary of Economics*, vol. 4, pp. 120 – 123, 1987.

[19] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[20] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *Proceedings of the Tenth National Conference on Artificial Intelligence*. MIT Press, 1992, pp. 223–228.

[21] H. Zhang, "The Optimality of Naive Bayes," in *Proceedings of the International Conference on Field Laser Applications in Industry and Research (FLAIR '04)*, V. Barr and Z. Markov, Eds. AAAI Press, 2004.

[22] J. B. MacQueen, "Some methods of classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability*, 1967, pp. 281–297.

[23] C. Chatfield, *The Analysis of Time Series: An Introduction, Sixth Edition (Texts in Statistical Science)*. Chapman & Hall/CRC, July 2003.

[24] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, 1996.

[25] B. Rood and M. J. Lewis, "Resource availability prediction for improved grid scheduling," in *Proceedings of the 2008 Fourth IEEE International Conference on eScience (e-Science 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 711–718.

[26] ——, "Scheduling on the grid via multi-state resource availability prediction," in *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (Grid 2008)*. IEEE, 2008, pp. 126–135.

[27] V. Pietrobon and S. Orlando, "Performance fault prediction models," University of Venice, Tech. Rep. CS-2004-3, 2004.

[28] H. Li, D. Groep, and L. Wolters, "Mining performance data for metascheduling decision support in the grid," *Future Gener. Comput. Syst.*, vol. 23, no. 1, pp. 92–99, 2007.

[29] W. Smith, I. T. Foster, and V. E. Taylor, "Predicting application run times with historical information," *J. Parallel Distrib. Comput.*, vol. 64, no. 9, pp. 1007–1016, 2004.

[30] K. C. Cho, T. Y. Kim, and J. S. Lee, "User demand prediction-based resource management model in grid computing environment," in *Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology (ICHIT '08)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 627–632.