

# Resource Fault Prediction Models

Valentino Pietrobon      Salvatore Orlando

Dipartimento di Informatica  
Università Ca' Foscari di Venezia  
Via Torino, 155 – 30172 Mestre-Venezia (Ve), Italy  
e-mail: {pietrobo|orlando}@dsi.unive.it

Technical Report CS-2004-3, May 2004

**Abstract.** The use of tools to forecast faults over computational resources composing highly distributed system, like Grid systems, represents an interesting approach to control the comprehensive performance of these systems, limiting the number of job submissions to those computational resources temporally unable to complete any job.

The analysis of log information retrieved from a real system shows an homogeneous behavior of the computational resources involved, and therefore the possibility to build forecasting tools limiting the number of job submissions that fail.

The results obtained in this paper show on one side the possibility to obtain high percentage of correct forecasts even with simple models, and from the other side the difficulty to improve these results even using adaptive models relatively complex.

**Keywords.** Computational and Data Grid Systems, Fault Prediction Models, Log Information Analysis

## 1 Introduction

To effectively exploit emerging platforms like Computational and Data Grids [7], it is needed to develop components to map and schedule user tasks over available resources, and to monitor resource performances. The availability of these system components are useful to improve resources utilization, taking into account user task requests (e.g. specific platform features, availability of given files, particular installed software, and so on), workload and temporal malfunction of the resources.

The experience on using computational resources indicates that a resource fault can compromise the effectiveness of resource selection and job scheduling. Faults can be caused by different situations: configuration problems, excessive workload, faults on the network, reboot of stopped resources, and so on. These situations produce temporal faults only over a subset of resources that evolve in unexpected way, and sometimes they do not completely compromise the performance of a given resource. In this situation, where the state of the resources changes frequently, it is impossible to use

usual tools, like Information Services, used to retrieve only the list of available resources at a certain time.

Therefore, in this paper we propose the development of a specific tool to forecast possible faults over a single resource, taking into account its past behavior. This solution requires the availability of information, tracing the past usage of every resource. Since, there exist different causes of a missed satisfaction for a given user computational request, these causes can be subdivided into classes and we can adopt a specific strategy for each specific class.

In this paper we take into account only fault prevention problem for computational resources used to execute user tasks. We want to avoid, when it is possible, the use of resources temporally not reliable. This result is obtained using log information from which we extract useful information regarding the behavior of the computational resources. Thus we build a suitable forecasting model about the real capability of resources to complete the execution of a job at a certain time. On the bases of these forecasts, we aim to map computational requests over resources limiting the number of tasks aborted during the execution.

This paper discusses fault forecasting models based on availability of resource log information functioning. We want to evaluate the possibility of foreseeing malfunctions of computational resources (or other resources) able to influence the correct execution of a job. Moreover, we want to evaluate a possible way to exploit the forecasts of resource malfunctions in order to improve the general performances of a real Grid system. In particular, we are interested in evaluating the real possibility to derive these forecasts, by exploiting information related to past elaborations, and to evaluate the real usability of such forecasts.

We show that the availability of log information on past executions suggests the possibility to get these forecasts and use them to choose what resources to use for satisfying the user requests. In fact, logs allow us to analyze the events of correct and failure job executions. Therefore, the availability of fresh log information is crucial.

If the past knowledge is fresh enough, a trivial model (that only uses the latest past information for a given computational resource) is enough to produce the right forecast. The limited advantages obtained by a more complex model do not justify the computational overhead needed to produce the prevision. Conversely, if log information is available only after a certain time period (e.g. 30 minutes) the trivial model loses part of its forecasting capability, and now complex models provide better previsions.

We analyze several models, derived from well known statistical models and adapted to the characteristics of the log information.

We also suggest some approaches to get some adaptive models able to evolve in the time and to produce results analogous to those obtained with the best static models (chosen looking at the whole results).

Finally we propose a possible architecture of a subsystem for faults predictions, suitable for the European Data Grid (EuDG) system.

The paper is organized as follows. Section 2 we recall some works present in literature. Section 3 describes the system used to retrieve log information used to perform the forecast of faults. In the same section, we describe the characteristics of log information, and we give some remarks on the feasibility of a forecast model inspired by this analysis. In Section 4, we describe the models used to forecast the future behavior of the computational resources using both a static version and an adaptive version of the forecast. The test results are discussed in Section 5, while in section 6, we introduce a possible architecture for a real Grid system. Finally, in the last section, we indicate other investigations and a possible different approach to solve this problem.

## **2 Related works**

In literature we found several works that propose the use of some parameters to describe the general performances of distributed systems or high distributed systems as the Grid systems.

Such works, consider both parameters to evaluate the computational performances of the computational unities [5, 6, 10], and the behavior and the performances of the network [1, 10].

The underlying idea is to be able to increase the general performances of the system, improving the capabilities of the scheduling policies adopted. The current proposed solutions use additional information on the past system evolution, described by past values associated with several system parameters, to forecast the future system behavior.

For example, we have systems for the monitoring of a distributed system like the Network Weather Service [10, 11, 12, 13], or several new scheduling strategies able to adopt adaptive techniques that use the actual state of the system [2, 3, 4].

On-line resource prediction systems like NWS collect measurements from the resources and use these measurements to predict future measurements, while the current adaptive techniques consider the state of the system described by the available parameters and the characteristics of the submitted jobs to decide where and in which order to submit the submitted jobs.

The extreme difficulty to check all the parts of a high distributed system and the frequent presence of unpredictable temporary malfunction regarding one or more resources, favorite the use of tools able to limit the negative effects of such malfunctions on the general performances of the system.

The production of fault tolerance system follows two main directions: the realization of robust resources and the adoption of scheduling strategies able to manage temporary breakdowns of one or more resources without compromising the global computational capability of the system.

From this point of view, the current Grid systems, like the EuDG [14], structurally characterized by resource faults, are developing scheduling politics able to manage temporary computational resource faults. This work gives a contribution in this direction.

### 3 Log Data

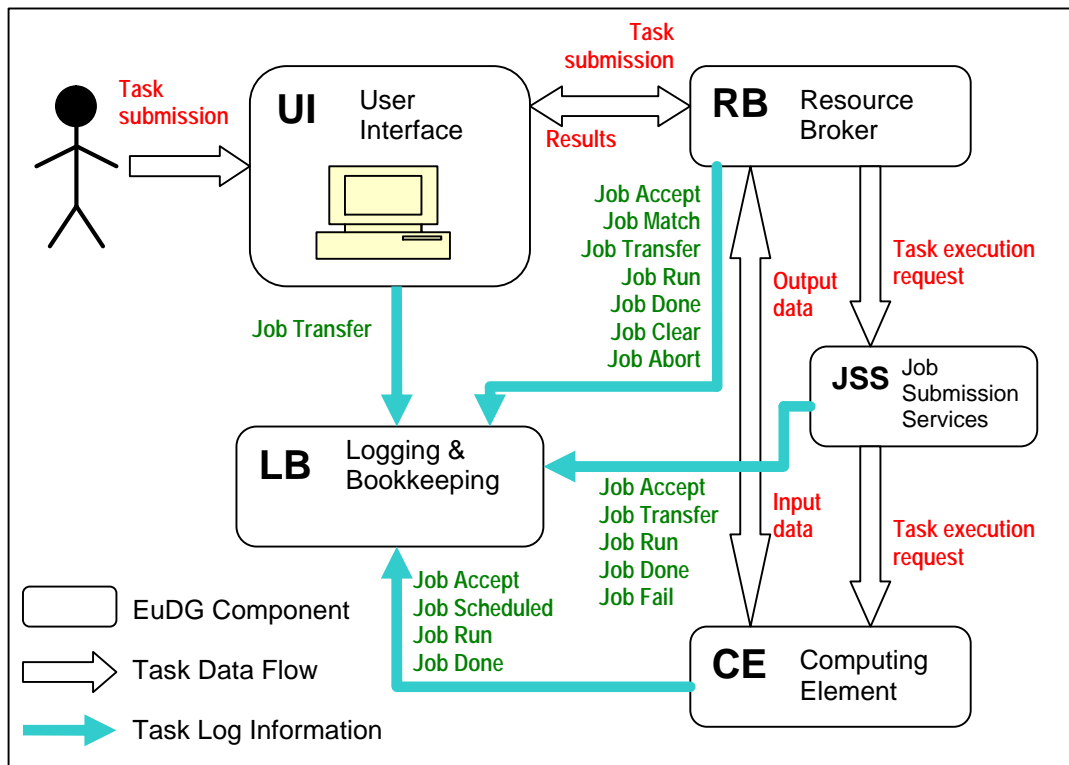
The availability of suitable log information to prevent faults in the computational resources (and in general for any class of resources) of highly distributed systems is not a trivial problem. In fact, varying amount, validity, accuracy and retrieval facility of these log information, it is possible to obtain solutions more or less effective using the same architectural model.

In our case, we use information available in the system EuDG Release 1.0. (a project finalized to develop a computational Grid oriented to handle data intensive tasks [14]).

Taking into account only the part of the EuDG Release 1.0 architecture, suitable for our purpose, we have (see Figure 1):

- a User Interface (UI) to provide a front end from which a generic user can submit his tasks to the Grid;
- a Resource Broker (RB) to decode the requests of the user tasks, to retrieve information about available resources needed to satisfy task requests and to decide, considering a proper scheduling policy, where execute the task;
- some Computer Element (CE) to execute user tasks and where operate the fault prevention;
- a Job Submission Service (JSS) to:
  - prepare the tasks that can be sent to the CEs for the execution,
  - guarantee the information retrieval needed for the task computation and
  - deliver the computational results;
- a Logging & Bookkeeping (LB) database, where all the message produced by the previous components are stored, and where it is possible to retrieve task information with proper “query” as: “retrieve the execution result for a given user task, or for a given CE”.

Since for every RB there is only, and only one UI, JSS and LB logically connected with a 1:1 relation, while many CEs are connected to the previous components with a 1:n relation, the system can be subdivided in more independent subsystems composed by a unique UI, a unique RB, a unique JSS, a unique LB and by many CE shared with others subsystems. In this architecture only CE are shared, even if in the real case only a part of there CE are actually shared.



**Figure 1 – The EuDG architecture including Task Data flow and log messages sent to LB**

### 3.1 Available information

The available log information was retrieved from 5 different LBs working in different periods partially overlapped, from May 2002 to the end of February 2003, and for a total amount of about 534 thousands log messages.

The quality of these messages is different because different was the status of the system and different is the number of available events for every LB. The period before November 2002 is not so relevant because there was only one LB activated and the system was used only to test its functionalities. Therefore, the log messages do not represent a real system behavior, while messages from January 2003 can be considered of a real system behavior.

Since, the system deployment is in progress, not all the LBs were active for the entire period. In fact, the first LB was shut down before the end of the 2002, and only in the 2003 all the other LBs was active.

Only the events coming from the third one and the fourth LB were sufficiently numerous and temporally dense to represent an effective situation of reality use of the system (see Table 1).

We subdivided the period of every LB into several sub-periods also considering together messages coming from different LBs if related to the same period. With these sub-periods we evaluated if the fault prevision became better considering the overlapping of different LBs' logs, and we evaluated if the behavior of a CE changed in the time suggests the use of adaptive models.

LB	# days	Execution done	Execution fallen	% Correct executions
1	96	1748	860	76.0%
2	109	1257	3437	26.8%
3	94	6448	12981	33.2%
4	45	8726	4465	66.2%
5	27	869	227	79.3%

**Table 1 – Correct and fail executions by period**

### 3.2 Log information filtering

For every user task, the EuDG system stores a number of log messages that varies, depending on the execution result. If no errors compromise the execution of a job, a few messages are generated by the system components, while, if one or more errors appear during the execution, the number of messages can become high. In fact, when an error appears, the system currently tries to resubmit the job to an available CE several times, thus generating several times messages describing submission and error occurrences.

Another reason, that influences the number of messages, is the loss of messages that never arrive to the destination.

Every log message has a common part (the initial 7 fields), also indicating the operation executed, and a variable part (from 1 to 4 fields), depending on the specific operation.

The redundancy of information in the log messages requires a preliminary filtering to extract synthetic information (it must be possible to extract information also when some messages are lost).

In more detail, we are interested not only on the relative information about correct or aborted elaborations, but we also want information on all the attempts done by the system in presence of faults before completing the execution.

In fact, all these information represent fault situations that must be considered to perform a reasonable forecast of errors for a given CE.

Finally, we take into account only fault situations produced by CEs discarding all faults produced by other components of the EuDG system like: RB, JSS or by the user. However, all faults due to communication problems (e.g. from a CE to RB/JSS, and vice versa) are considered as CE faults by the EuDG system.

From the filtering of the available messages we obtained a set of events containing the following information:

- the job identifier
- the CE selected to execute a given job
- the result of the execution

- the first time instant from which it is possible to understand if a given execution is correctly finished or an error appears
- the possible error message

Analyzing the obtained events, we can see that the percentage of correct executions sometimes is low. This suggests the need to prevent faults at the CE level (see Table 1).

### 3.3 Data analysis

The considered events describe essentially two opposite situations:

- a given CE that correctly executed a job: we assume the CE is perfectly working;
- a given CE that did not execute a correct job (without user errors): we assume the CE is temporally breakdown.

Since a CE is commonly composed of several resources (generally a farm with many Working Nodes “WNs”, and a queue for jobs that are waiting the first WN free able to accept the job and execute it), it is reasonable to suppose that the failure can involve:

- a subset of WNs: in this case , the job can be correctly executed also reselecting the same CE;
- a fundamental resource for the execution of the job (e.g. the queue): in this different case, if we resubmit the job to the same CE, the result does not change;
- the communication between CE and RB/JSS: in this last case, if we resubmit the job to the same CE we could obtain any result. But, from a RB/JSS point of view, this situation is the same as the case when the whole CE is unreliable.

Therefore, having to take into account all these possible scenarios, it is important to understand if the behavior of a given CE is characterized by a temporal persistence. In fact, if this property holds, then forecasting the future CE behavior become now effective.

In order to verify this important property, we take into account all the possible couples obtained considering together an event and any other event happened before it in the same CE.

With  $E0_{\Delta t}^0$  and  $E1_{\Delta t}^0$  we denoted a correct execution happened after a correct execution, and an aborted execution happened after a correct execution. While, with  $E0_{\Delta t}^1$  and  $E1_{\Delta t}^1$  we denote a correct execution happened after an aborted execution, and an aborted execution happened after an aborted execution. The parameter  $\Delta t$  indicates the time between the two considered events.

With  $E0$  and  $E1$  we simply indicate a correct execution and an aborted execution.

We have a temporal persistence of the CE’s behavior if:

$$P(x = 0) < P(x = 0 | y = 0) \tag{3-1}$$

and

$$P(x = 1) < P(x = 1 | y = 1), \tag{3-2}$$

where  $y$  indicate an event happened before the event  $x$ .

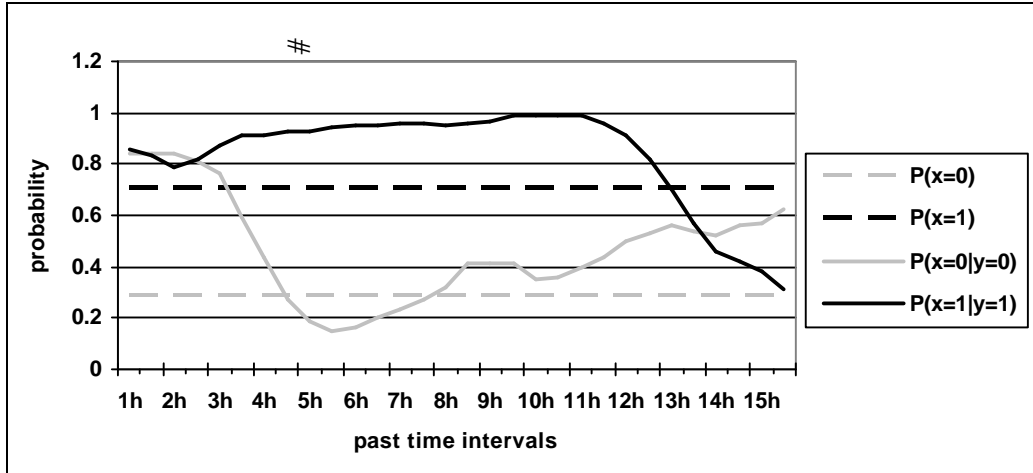
But if the number of considered events is big, we have:

$$P(x=0) \simeq \frac{\# E0}{\# E0 + E1} < P(x=0 | y=0) \simeq \frac{\# E0_{\Delta t}^0}{E0_{\Delta t}^0 + E1_{\Delta t}^0} \quad [3-3]$$

and

$$P(x=1) \simeq \frac{\# E1}{\# E0 + E1} < P(x=1 | y=1) \simeq \frac{\# E1_{\Delta t}^1}{E0_{\Delta t}^1 + E1_{\Delta t}^1}. \quad [3-4]$$

But varying  $\Delta t$  over a continuous past time interval, we have not enough couples of events to calculate these conditioned probabilities for every value of  $\Delta t$ . Therefore, instead of considering all the possible values of  $\Delta t$ , we subdivided the past time interval into subintervals big enough (in our case 30 minutes) to obtain a reasonable numbers of couples in everyone of them.



**Figure 2 – Example of probabilities conditioned to past events belonging class 0 and class 1 for the CE testbed008.cnaf.infn.it:2119/jobmanager-pbs-short over past intervals of 30 minutes.**

The behavior of these conditioned probabilities altogether shows a temporal persistence that decreases with the increase of the time distance (see Figure 2), and often the conditioned probability touches the line of the not conditioned probability (point over which every past event can not help anyway to forecast a given event).

## 4 Forecasting models

In this paper some forecasting models are proposed. These models result to be, on one side, a generalization of the classical averages of past values taken over a fixed-length history, and from the other side, a restriction of the autoregressive models.

The classical model obtained as the average of all the past  $t$  values, defined by:

$$AVG(t) = \frac{1}{t} \sum_{i=1}^t value(i), \quad [4-1]$$

does not work, because past events too much away from the event to forecast do not improve the prevision, and sometimes introduce dangerous noise.

Thus, it is essential to use a “sliding window” average model ( $SW_{AVG}$ ), calculated as:

$$SW_{AVR}(t, k) = \frac{1}{k} \sum_{i=t-k-1}^t value(i), \quad [4-2]$$

where only the most recent  $k$  values are used. This approach is important to forecast new unexpected situation described only by the latest events.

The autoregressive models [9], differently from the “sliding window” average models, weigh past values according to the temporal distance between the value to foresee and the known value. The weights are typically inversely proportional to this distance, but they can generally assume any configuration, thus implementing a complex weighting.

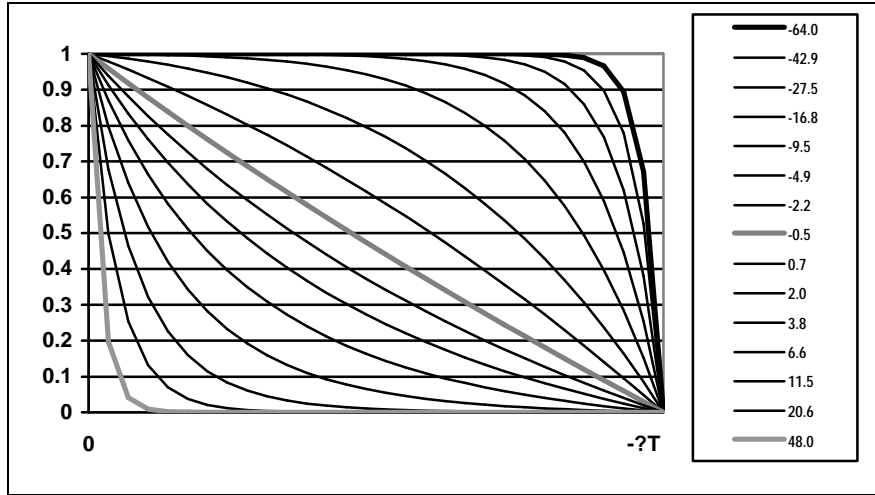
The general form of a  $p^{\text{th}}$ -order autoregressive model is

$$AR(t, p) = \sum_{i=0}^p a_i \cdot value(t-i). \quad [4-3]$$

Unfortunately the autoregressive models use time series of values registered at regular time intervals  $t, t+?T, t+2?T, t+3?T, t+4?T, \dots$ , weighted by the coefficients  $a_i$ , while in our case the past values may refer to any time instant. For this reason, instead of calculating representative events from the known real events with a given refresh rate, we decided to use less effective models, but easily applicable to our past events, and whose weight are given by

$$a_i(t_i, b, \Delta T) = \frac{c_i(t_i, b, \Delta T)}{\sum_{k=1}^N c_k(t_k, b, \Delta T)}. \quad [4-4]$$

The parameter  $t_i$  is the time distance between the event to foresee and the past value to weight with  $a_i$ ; the parameter  $b$  is used to define the weights in the time (see Figures 3). The values assigned to  $b$  produce different shapes for the weights: a big positive value ( $b > 40$ ) gives much more importance to the latest past events (similarly to the trivial models illustrated in the following), while a big negative value ( $b < -40$ ) associate a similar weight with all the considered past events in the  $?T$  interval. The parameter  $?T$  identifies the length of the past time interval, while  $N$  identifies the number of past events occurring in the interval  $?T$ . The denominator normalizes the weights  $a_i$  to the real interval  $[0,1]$ .



**Figure 3 – Parameters  $c_i(t_i, b, \Delta T)$  as a function of the parameter  $t_i$ , where each curve is relative to a different parameter  $b$ , ranging from -64 to +48.**

The coefficients  $c_i$  are given by:

$$c_i(t_i, b, \Delta T) = \frac{M - \frac{1}{\left(1 + \frac{t_i}{\Delta T}\right)^b}}{M - m}, \quad [4-5]$$

$M$  and  $m$  are defined as:

$$M = \max_{t_i \in [-\Delta T, 0]} \frac{1}{\left(1 + \frac{t_i}{\Delta T}\right)^b} = \max\left(1, \frac{1}{2^b}\right) \quad [4-6]$$

and

$$m = \min_{t_i \in [-\Delta T, 0]} \frac{1}{\left(1 + \frac{t_i}{\Delta T}\right)^b} = \min\left(1, \frac{1}{2^b}\right). \quad [4-7]$$

## 4.1 Static parametric models

We considered both static models and adaptive models. Adopting a static model we fix a priori the parameters of the model to use during the forecasting process, while with adaptive models it is possible to modify such parameters dynamically, on the basis of the behavior of the model in correctly forecasting events.

For testing the static models, we considered two different cases. In the first one each past event considered correspond to a single job submission, while in the second we divided past events into fixed length subintervals, calculated their averages for every subinterval, and used such averages as past events to forecast a new event. The choice to also consider averages of event intervals derives from the observation that a submissions that fails also produces, after a small time delay, a new submission (often to the same CE), and such submission usually fails again. So, it can be useful to filter such events that represent the same overall situation of the system.

The analytical form of the first type of models (Single Values Model- SVM), that uses single past values, is given by:

$$SVM(t_0, b, \Delta T) = \sum_{i \in \{j | t_j \in [t_0 - \Delta T, t_0]\}} a_i(t_0 - t_i, b, \Delta T) \cdot value(i), \quad [4-8]$$

where  $value(i)$  is the past value occurred at the time instant  $t_i$ , while  $t_0$  is time instant of the event to foresee.

The analytical form of the second type of models (Mean Interval Values Model – MIVM) is defined as:

$$MIVM(t_0, b, \Delta t, \Delta T) = \sum_{i \in \left\{j \left| t_j \in \left[ t_0 - \frac{1}{2}\Delta t, t_0 - \frac{3}{2}\Delta t, t_0 - \frac{5}{2}\Delta t, \dots \right] \right\}} a_i(t_0 - t_i, b, \Delta T) \cdot \overline{value}(i), \quad [4-9]$$

where  $\Delta t$  is the time length used to subdivide  $\Delta T$  into sub-intervals and  $t_i$  is the middle time instant of the sub-intervals.

$\overline{value}(i)$  defined in this way:

$$\overline{value}(i) = \frac{\sum_{l \in \{k | t_k \in [t_i - \frac{1}{2}\Delta t, t_i + \frac{1}{2}\Delta t]\}} value(l)}{\left| \left\{ k \mid t_k \in \left[ t_i - \frac{1}{2}\Delta t, t_i + \frac{1}{2}\Delta t \right] \right\} \right|}, \quad [4-10]$$

is the average value of the events in the sub-interval  $i$ .

A model of the first family is comparable with one of the second family when:

$$\Delta T = N \cdot \Delta t, \quad \text{con } N = \lceil \Delta T / \Delta t \rceil. \quad [4-11]$$

The models [4-9] and [4-10] have been generalized to use them in a real context. In fact, the retrieval of events from a generic LB is not instantaneous, since it depends on the overload of the LB and the time needed for the information communication.

For our system it seems reasonable to retrieve the past events every ten-thirty minutes.

Moreover, there is also the necessity to ignore the events happened too much recently and not yet available (because of the information propagation time delays in the system and the computational time needed to satisfy the information request by the LB). We thus introduced a new parameter  $t_d$  that

represents the time distance between the value to foresee and the beginning of the time interval of available past events. Considering an update made every 30 minutes and a delay of one minute to deliver the log information, we obtain a  $t_d$  that varies from 1 minute to 31 minutes.

The definition of the first class of models becomes:

$$MSV_{SD}(t_0, t_d, b, \Delta T) = \sum_{i \in \{j | t_j \in [t_0 - t_d - \Delta T, t_0 - t_d]\}} a_i(t_0 - t_d - t_i, b, \Delta T - t_d) \cdot value(i), \quad [4-12]$$

while the definition of the second class of models becomes:

$$MIVM_{SD}(t_0, t_d, b, \Delta t, \Delta T) = \sum_{i \in \left\{j \left| t_j = t_0 - t_d - \frac{1}{2}\Delta t, t_0 - t_d - \frac{3}{2}\Delta t, t_0 - t_d - \frac{5}{2}\Delta t, \dots \right.\right\}} a_i(t_0 - t_d - t_i, b, \Delta T) \cdot \overline{value(i)}. \quad [4-13]$$

When none past values is present in the interval  $[t_0 - t_d - \Delta T, t_0 - t_d]$ , we consider the last known past events.

Nearby the discretization of the parameter  $b$ , we also considered a discretization for the parameters  $\Delta t$  (the time length of the sub-intervals that subdivide  $\Delta T$ ) and a discretization for parameter  $\Delta T$  (size of the past interval used for the forecast). Therefore instead of considering infinite values for these parameters, we considered finite sets of values.

Since we wanted a forecast characterized by only two classes: 0 (correct computation) and 1 (malfunction), if the forecast was smaller of 0.5, it was labeled with 0, while if it was greater or equal to 0.5, it was labeled with 1.

We also determined what family of models and what parameters guaranteed the best forecast capability; this process allowed us to determinate at the end of the prevision and for every considered period of time the maximum percentage of correct forecasts

Finally, we evaluated some trivial models where the forecast was directly obtained by the last available past event, or as the average of the values of the available last past events that fall in the nearest sub-interval  $\Delta t$ .

## 4.2 Adaptive models

In several cases, the considered static models cannot be applied, because, they have none adaptation capability to the behavioral evolution of a given computing element. They are based on an a priori decision about the parameters to use, that can be verified only at the end of the previsions, looking at the obtained results. Therefore, it is impossible to decide before every prevision which values to assign to these parameters in order to improve the total number of events correctly forecasted.

Therefore, it becomes necessary to explore possible adaptive solutions able to dynamically select the parameters of the models, dynamically choosing the most promising ones.

In order to realize this exploration, we considered two particular aspects:

- the length of the past time interval  $DPT$  used to evaluate the current capability of the model;
- the way to switch from a given combination of parameters to the most promising ones.

For the past time interval  $DPT$ , we evaluated 2 opposite solutions:

- $\Delta PT = ]-\infty, t_L]$  (we compare all the past previsions with the correspondent past events obtained from the beginning of the activity of the considered computing element to the  $t_L$  time instant);
- $\Delta PT = [t_L - \Delta tu, t_L]$  (we compare the past previsions with the correspondent past events falling only in the last update time interval  $Dtu$ );

With  $t_L$  we indicate the ending time instant of the last update time interval  $Dtu$ . The update time interval  $Dtu$  is defined as the period between an event update (the time instant when arrive new log events from a given LB) and the previous one.

The choice of the values to assign to the model's parameters is based on different strategies considering events ad obtained previsions for the time interval  $DPT$ .

Every strategy first defines the parameter  $DT$ , then the parameter  $Dt$  and finally the parameter  $b$ , but differs from the others on the speed to pass from a set of values to a new set of values for the parameters of the adaptive model. For instance, some strategies ignore possible oscillations of these values assigning to the parameters the values that produce the best previsions in the time interval  $DPT$ , while other strategies limit this possibility incrementing o decrementing the values of one unit.

## 5 Results

We evaluated the adaptive and static models with two sets of tests.

- With the first set of tests we compared results obtained with static models in different situations in order to evaluate the intrinsic characteristics of the events and the best set of values to assign to the models' parameters. We also compared static models and trivial models in order to evaluate the potential forecasting capability of the first ones.
- With the last set of tests we used several adaptive models characterized by different adaptive strategy, in order to identify the best strategy to select the most promising model's parameters on the basis of the known past events.

For all the tests the events when retrieved from the LBs (events update) at regular time intervals, and considering a delay of one minute from the RB request to the associate LB, and effective availability of the retrieved events to perform the forecast.

In all the analyzed situations we found a progressive percentage reduction of the correct previsions (also some points in percentage) with respect to the progressive time increment between an events update and the next events update.

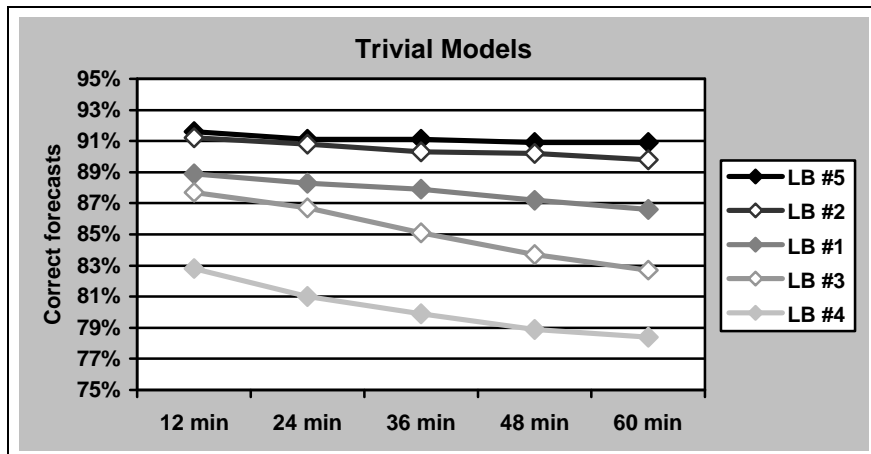
With the first set of tests we evaluated the possible percentage improvement of correct forecast obtainable by:

- using specific static models for every single CE instead of a common model for all the CEs;
- using together events retrieved from different LBs (for the same CE and the same period) instead of using events coming only from one LB at a time;
- using several different sets of values assigned to the parameters at different times;
- using the static models instead of the trivial models.

The improvements obtained by differentiating the parameters' values, by considering for every single CE, did not result so relevant: less than half point percentage of improvement for three LBs (with few events) and only one point percentage of improvement for the other two LBs (with more events). In many case the choice of models and parameters were not so relevant.

We also found that the contemporary use of events, coming from several LBs, did not produce concrete advantages, but sometimes produced a mean percentage decrement of half point on the total number of correct forecasts.

Then, we found dependence between the percentage of correct forecasts and the discretization of values assigned to the parameters of models. But the percentage improvement obtainable with a refinement of the discretization is limited (a mean increment of three decimal point percentage). Therefore, a new refinement is not needed.



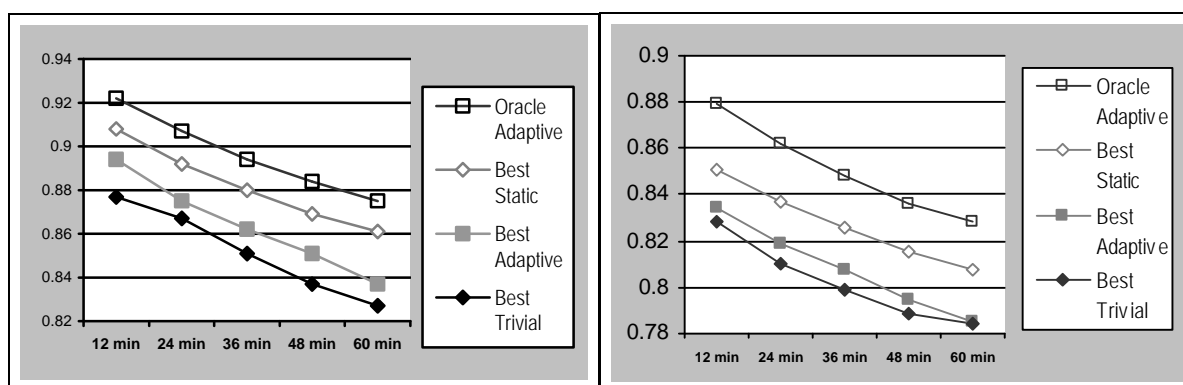
**Figure 4 – Mean percentages of the correct forecasts obtained with the best trivial model for all the five considered LBs, as a function of the time interval length from two consecutive events update.**

All the considered trivial models (using as forecast the last value available, the average of the events of the last minute, and the average of the events of last five minutes) show the same behavior as the same results. In Figure 4 we can see the mean percentages of the correct forecasts obtained with the best trivial model for every LB, as a function of the time interval length between an events update and the next one.

The percentage improvement of correct forecasts obtained by the best static models with respect to the best considered trivial models, varies only from 1 and 3.5 points of percentage, considering, however, that the percentages of correct forecasts obtained with the trivial models are already high (80-90% - see Figure 4).

Finally, we evaluated the capabilities of the considered adaptive models by analyzing:

- several adaptive strategy to adapt the values of the models' parameters in the time;
- the predictive capability of these models with respect to the trivial models, the best static models and the adaptive models with oracle (these models that always know the best value to assign to the parameters, give a the maximum theoretical percentage improvement of correct forecast obtainable with the adaptive models).



**Figure 5 - Mean percentages of the correct forecasts obtained with the best of the considered trivial model, the best of the considered adaptive model, the best of the considered static model and the adaptive model with oracle, for events retrieved from the third LB (left graph) and forth LB (right graph).**

All the considered adaptive strategies did not give suitable results: they produced only a mean percentage improvement of the correct forecasts of one point percentage with respect to the trivial models (third and forth LB); improvements inferior respect to those obtained with the static models.

The maximum percentage improvement (also of 3-4 points percentage) was obtained only with the adaptive models with oracle. Obviously, it is impossible to define an effective adaptive strategy able to select the best parameters when the CE's behavior can change unexpectedly. Anyway the adaptive models with oracle give an upper bound for the maximum obtainable percentage of correct forecasts (see Figure 5).

Therefore, it seems not so much useful to use complex forecasting models instead of a more simple trivial models that require less computational resources and memory and they offer however good percentages of correct forecasts. From this point of view, it seems decisive to have a simple and fast procedure to retrieve past events from LB, to be refreshed as often as possible.

## 6 A proposed architecture

We synthetically give some indications of a possible architecture that implements a mechanism to limit the number of submissions to resources that temporally fail.

### 6.1 Requirements

From a simple analysis of the percentage of job submission correctly executed, made for the available LBs (see Table 1), we can see the necessity to have a tool able to do “fault prevention” or in any case at least able to do “fault prediction”. In fact, the percentages of jobs correctly executed result sometimes low, and produces system overhead and low throughput.

Nevertheless, the percentage of executions forecasted in a correct way by the considered models (also the trivial models) is greater than the percentage of job submissions executed in correct way.

Therefore it is desirable an automatic tools able to forecast faults of computational resources and able to help the scheduler to decide which computational resource select to execute a given job, in order to increase the percentage of job executed without faults.

It is however important to remember that, in some particular situations, a computational resource can not be completely ignored even if a fault prediction is obtained. For instance, if there is only a resource able to perform a given job, or if the scheduler does not choose the computational resource only considering the fault forecasts, we want to continue to select this computational resource.

This behavior is not only acceptable but it guarantees the possibility to reuse a given resource after a period of malfunction. In fact, if a resource is not used, none new event will certificate its correct functioning and this resource will not used any more.

### 6.2 The architecture

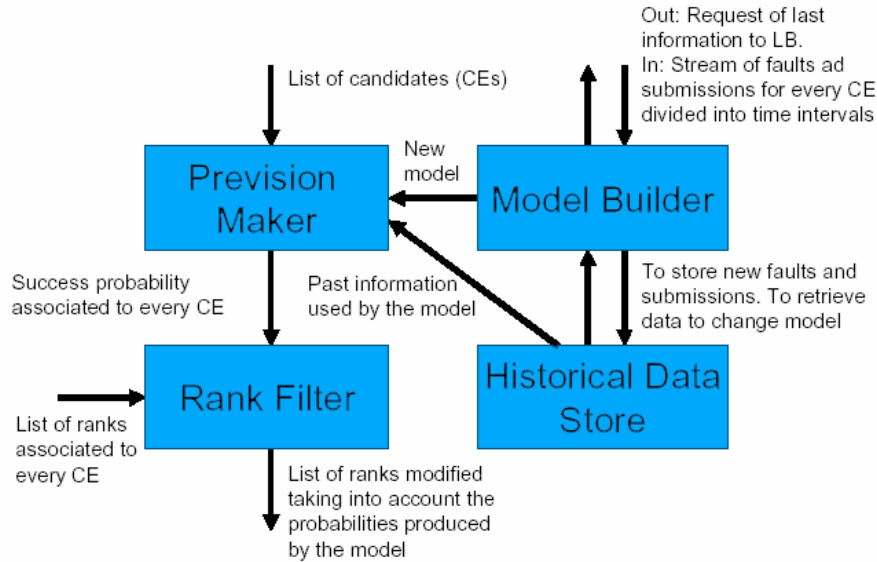
When a job submission fails, the RB identifies the fault situation and proceeds to resubmit the job to another CE. This operation takes time because it is necessary to determinate the new best CE to execute job, to send the job to the selected CE and to wait in the CE queue that previous jobs are executed.

So using log-information and a specific tool we want to foresee faults penalizing the use of CEs, e.g., by modifying ranks used to determinate the list of CE that satisfy the job request.

This tool that we will name Resource Fault Prediction Model (RFPM) (see Figure 6), consists of 4 logical parts:

- **The Model Builder:** It requires log information from the LB concerning failures and successes at regular time intervals. These log information have to be filtered to discard redundant and bad information. Then, it builds a model using historical data stored in the Historical Data Store. Finally, it updates the current model according to new information retrieved from the LB when the

forecasting capability of the current model decreases under a reasonable level, or it can be increased.



**Figure 6 : Proposed architecture for the Resource Fault Prediction Model**

- **Historical Data Store.** It stores the relevant past information used to modify the model and to calculate the fault probability associated with a give CE. Only failures and successes occurred in the last interval time  $T$  are stored and the others are discarded.
- **Prevision Maker.** It calculates the fault probability associated with all the CEs that satisfy the job requests considering past data stored in the Historical Data Store, and the model provided by the Model Builder.
- **Rank Filter.** It calculates the new order of the CEs satisfying job requests, by also taking into account the ranks specified by the user and the evaluated probability of fault.

## 7 Conclusions and Future Work

The analysis made in this work shows, on one side, the necessity to have a tool able to limit the job submission to computational resources temporally spoiled, and, from the other side, the feasibility of such tool.

Excellent results were obtained with trivial models, but problems happened to foresee unexpected changes in the behavior of the considered computational resources. We obtained a limited increase of the percentage of correct forecasts using adaptive models with respect to the use of trivial models. By using adaptive models with oracle we obtained an interesting percentage improvement of correct forecasts (from 1 point percentage to 4 points percentages) representing the upper bound obtainable with these models.

In this work we also presented a practical approach to manage events that happen “randomly” in the time; we thus proposed to use models more complex than the classic “sliding window” average models, but less complex than the autoregressive ones.

However, new experiments using new logs could be useful, if these logs represent a more realistic situation of the Grid system usage.

It is also important a further exploration of the adaptive models to obtain an improvement of their forecasting capability, for example considering other information that describes the evolution of the system (e.g. the load of the system) or considering other adaptive strategy.

It could be also interesting to investigate the possible use of approaches adopting past representative events calculated at regular time intervals. In this case we can use standard approaches and available tools [11] that implement a lot of models like autoregressive models. This approach needs a way to convert our events happening at random time instants into equivalent measurements calculated at regular time intervals and describing the real instant capability of a computational resource to execute a job without faults.

Anyway, an evaluation of the effective improvement of submission completed without faults for a real system using the proposed architecture is also needed. In a real situation, we have to decide the weight to assign to the fault prevision inside the algorithm used to select the computing element that normally it depends also from others factors

## References

- [1] S. Basu and A. Mukherjee, "Time Series Models for Internet Traffic," in 24th Conf. on Local Computer Networks, Oct. 1999, pp. 164--171.
- [2] F. Berman and R. Wolski. Scheduling from the perspective of the application. In Proceedings of the Fifth IEEE Symposium on High Performance Distributed Computing HPDC96, pages 100--111, August 1996.
- [3] H. Casanova, M. Kim, J. Plank, and J., Dongarra, Adaptive Scheduling for Task Farming with Grid Middleware, The International Journal of High Performance Computing, Vol. 13, No. 3, Fall, 1999.
- [4] H. Casanova, G. Obertelli an F. Berman, and R. Wolski. The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid. In Proc. of Super Computing 2002, Dallas, Texas, USA, November 2002.
- [5] P. A. Dinda and D. R. O'Hallaron. An extensible toolkit for resource prediction in distributed systems. Technical Report CMU-CS-99-138, School of Computer Science, Carnegie Mellon University, July 1999.

- [6] P. A. Dinda and D. R. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction," presented at Proceedings of the 8th IEEE International Symposium on High-Performance Distributed Computing (HPDC-8), Redondo Beach, CA, 1999.
- [7] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufmann, 1999.
- [8] L. M. Silva, B. Veer, and J. G. Silva. How to get a fault tolerant farm. In World Transputer Congress, pages 923-- 938, Aachen, Germany, September 1993.
- [9] N. N. Tran. Automatic ARIMA Time Series Modeling and Forecasting for Adaptive Input/Output Prefetching. PhD thesis, University of Illinois at Urbana-Champaign, Dec. 2001.
- [10] S. Vazhkudai and J. Schopf. Predicting sporadic grid data transfers. In Proceedings 11th IEEE Symposium on High Performance Distributed Computing (to appear) , July 2002.
- [11] R. Wolski, "Dynamically forecasting network performance using the network weather service," Journal of Cluster Computing, Vol. 1, No. 1, 1998, pp. 119-132.
- [12] R. Wolski, "Dynamically forecasting network performance to support dynamic scheduling using the Network Weather Service". In 6 High-Performance Distributed Computing Conference, August 1997.
- [13] R. Wolski, N. Spring, and C. Peterson. Implementing a performance forecasting system for metacomputing: The Network Weather Service. In Proceedings of Supercomputing Conference, November 1997. 8, 26, 26 105
- [14] The DataGrid Project. <http://www.eu-datagrid.org/>
- [15] NWS. <http://nws.cs.ucsb.edu/>
- [16] Globus. <http://www.globus.org/>