

# Prediction $f$ based Models for Evaluating Backfilling Scheduling Policies

F. Guim<sup>1</sup>, J. Corbalan<sup>2</sup> and J. Labarta<sup>3</sup>

*Barcelona Supercomputing Center*

Jordi Girona 1-3, Barcelona, 08034, Spain

<sup>1</sup>francesc.guim@bsc.es

<sup>2</sup>julita.corbalan@bsc.es <sup>3</sup>jesus.labarta@bsc.es

## Abstract

*The research on the usage of prediction techniques in HPC scheduling policies rather than user estimates has increased its relevance these recent years. In the coming scheduling architectures, like Grids and very heterogeneous computational resources, such techniques are having a crucial relevance due to users in most of the cases will not have enough information or enough skills for specify for how long will their jobs run.*

*Many studies have analyzed the impact of the user runtime estimates accuracy in the performance of the scheduling policies. Using user runtime estimation models, such as the  $f$ -model, researchers have evaluated how the accuracy of the runtime estimates provided by the user at the job submission can affect the performance of the backfilling policies and its variants. However, these traditional estimation models can not be applied to backfilling scheduling policies that use runtime predictions rather than user estimates. Clearly, predictions can not be characterized with these models. For instance because the underestimation of the runtime is not considered by them and obviously it can occur.*

*In this paper we describe and evaluate a set of  $f$ -model based prediction models that characterize the behavior that prediction techniques have shown in HPC centers. They have been designed for evaluate scheduling policies that use predictions rather than user estimates.*

## 1 Introduction & Motivation

Backfilling based scheduling policies have been demonstrated to be the scheduling policies that achieve higher performance in parallel HPC architectures. However, the major inconvenience that their base algorithm has is that the scheduling is based on the job runtime estimation that users provide. Thus, the runtime of the scheduled applications

has to be known at submission time.

Several studies have carried out extend research on the impact of the user estimates accuracy in the performance of these scheduling policies. The commonly used  $f$ -model (introduced later) has been used for evaluate how the accuracy of the user estimations affect the performance of the backfilling and its variants. Thereby, using such analysis researchers have proposed new backfilling variants scheduling policies, like S-H Chiang did with LXWF-Backfilling or Tsafirir did with Shortest Job Backfilled First (SJB-First).

The topic that has increased its relevance these last years is the usage of prediction techniques in scheduling policies rather than user estimates. In the coming scheduling architectures, like Grids and very heterogeneous computational resources, such techniques are having a crucial relevance due to user in most of the cases will not have enough information or enough skills for specify how long it will his/her job run. Some researches have started to face this complex problem. Tsafirir et al. have provided to the community two works studying the impact of the usage of four different prediction techniques rather than user estimates in the backfilling policies [21]. Similar work is the once presented by Talbi et al. in [19]. They evaluate the impact of the usage of three different predictors for the backfilling policies that are designed to work with different amount of information at the prediction time. All these works have analyzed the impact of specific predictors with specific properties to these backfilling policies.

The traditional  $f$ -model can not be used in the evaluation of the performance of scheduling policies that use prediction techniques rather than user estimates. As will be discussed later, this model does not properly model some characteristic inherent to the prediction techniques that can cause important effects to the behavior and performance of the studied scheduling systems. This is mainly caused due to the essence of the predictor properties are clearly different for the ones that users usually show in their runtime estimations. For example, while users usually tend to over-

estimate their jobs with a factor of 2, or even 5, for avoiding their jobs to be killed, a predictor can underestimate the job runtime with a factor of 100. Obviously this is because of real users have some hints concerning how they applications will behave while predictor does not. In this paper we have defined a set models having several properties inherent to the predictors estimations that can have impact to the performance of the system. This characterization has been based on our research background in prediction and scheduling techniques [7][10][11][14][9][12].

We have analyzed the impact of these models to different set of workloads available in [5]. In the experiments we have stated that adding or subtracting quantitative errors to all the runtime jobs predictions of the workload have no relevant impact on the performance of the system. Furthermore, adding/subtracting quantitative errors to a determined subset of jobs does has not relevant impact in the performance of the system. The unique subset of jobs that has shown a real impact in the performance has been the short jobs. We have also demonstrated how qualitative errors on the runtime predictions can result in a dramatic drop of the performance. Thus, predicting that a job will be short while it is really large or vice versa can substantially increment performance variables such as the average bounded slowdown.

The rest of the paper is organized as follows: sections 2 and 3 we present the related work and our main contributions; sections 4 and 5 the simulation environment used and the prediction models are described; sections 6 and 7 the experiments and the evaluation of the models are presented; and finally, in section 8 we present the conclusions.

## 2 Related Work

The backfilling [16] policies have been the main goal of study these recent years. Authors like Franchtenberg or Chiang have provided to the community many quality works regarding to this topic. In [6] general descriptions about the most used backfilling variants and parallel scheduling policies are presented. Moreover, deeper description of the conservative backfilling algorithm can be found in [18], where the authors present it characterization and how the priorities that can be used when choosing the appropriate job to be scheduled.

Regarding the usage of prediction models in backfilled based scheduling policies, some authors have evaluated the usage of specific predictors rather the user runtime estimates. At the best of our knowledge, Tsafirir et al. described the first work in this topic [20] in this type of policies. The authors not only evaluated the impact of using prediction techniques, they also described the needed extensions in the scheduling policies for support such usage. Following the same approach, Talbi, also from the Uni-

versity of Jerusalem, provided a performance analysis of the impact of using the Session-based, estimation-less, and information-less runtime prediction algorithms in the Backfilling scheduling policies. The research activity presented in this paper has been mostly based on the description that both authors propose.

More extensive literature is available in the area of performance prediction. The usage of time series is proposed in several works, for example Peter A. Dinda in [4] propose the usage of linear mathematical models for predicting the runtime of applications. Similar to the techniques presented by Dinda, Yuanyuan [22] introduces new models also based in time series. In [3] the authors proposed the utilization of a state-transition model to characterize the resource usage of each program in its past executions. The other frequently used approach is using simulations for predicting the job runtime. For example, the PACE simulator [13] that is an object-oriented simulator and execution system based on high-level stochastically Petri nets with time modeling. Finally the last group of prediction techniques that has been recently used by several works are the datamining techniques. Relevant examples are the once carried by Warren Smith et al. in [17] or Gibbons in [8] using clustering techniques. More recent work is the once presented by Hui Lui [15], where they presented a new prediction for wait queue time and runtime using the K-Nearest Neighbor technique.

## 3 Contribution

In this paper we present a detailed study of which is the impact of the prediction errors in the scheduling policies. Clearly, our work has focused on several issues that have not been modeled in the other works. The traditional user runtime *f-model* used in the backfilling scheduling policies evaluations is defined as follows:  $est_{\sigma} \in U [r, r * \beta]$ . However, as we have introduced before it does not consider some properties that predictions have. In the presented evaluation work we have characterized a set prediction models that extend the *f-model*:

- We have used models where the prediction of the runtime can be lower than the real runtime. The estimation *f-model* does not take into account that jobs can be underestimated due to jobs would be killed by the schedulers.
- The other works have used simple error models, since the experiments the same estimation accuracy model is applied to all the jobs. We have evaluated different profiles of errors. By profile, we understand to apply errors a jobs with a certain characteristic, for instance: those jobs that use executables that are more frequently submitted to the center or those jobs that, respect to the

workload used in the simulation, requires a high number of processors etc. Here, we have studied the potential use of specific predictors specialized in a certain job typologies.

- We have evaluated the impact of changing the nature of the job with the prediction using qualitative errors. For example, a qualitative error is to predict that the job runtime of the job is short while, respect to the simulated workload, the job was really large. In this scenario, we also have analyzed the potential use of categorical predictors in the context of the backfilling policies.
- As proposed in [21], we have extended the backfilling interfaces for treat the deadline missed event for a job. This is triggered when a job has used all the runtime that was predicted when it was submitted. In these situations, differently from the approaches that are taken in the traditional studies, the job can not be killed due to the error has not been generated by the user. In our work what the scheduler does is extending the prediction and the job allocation with an amount  $\alpha$  of time.

## 4 Simulation Framework

All the experiments presented in this paper have been conducted using a C++ event-driven simulator that was implemented and used by Tsafirir et al. in the paper [21]. This is a modular simulator that allows to simulate the EASY-Backfilling and SJB-First policies. They are implemented extending the traditional EASY algorithm for the usage of runtime prediction rather than user estimates. It also allows adding predictors modules that are used by the scheduling policies for schedule the job according its predicted runtime. For the work presented in this paper we have extended the simulator implementing the other backfilling variants and implementing the *fake predictor*. It predicts the job runtime carrying out numerical transformations on the real runtime (presented in section 5). We called it ‘fake’ due to its predictions are computed by adding or subtracting a value to the real runtime of the job taken from the workload.

## 5 The Prediction Models

The models that have been characterized are divided in two main groups: quantitative prediction errors and qualitative prediction errors.

### 5.1 Prediction Generation Using Quantitative Errors

The quantitative errors are generated by adding or subtracting a  $\alpha$  percentage to the real runtime of a job and re-

Workload	RT boundaries <sup>1</sup>		%jobs
	Lower bound	Upper bound	
l_lanl_cm5_cln	Lower bound	224.27	50% shorts
	Upper bound	5606	20% larges
l_sdsc_par95	Lower bound	609	80% shorts
	Upper bound	15240	10% larges
l_sdsc_par96	Lower bound	621.34	65% shorts
	Upper bound	15534	%15 larges
l_kth_sp2	Lower bound	956.88	55% shorts
	Upper bound	23922	15% larges
l_ctc_sp2	Lower bound	1152.2	55% shorts
	Upper bound	28804	20% larges

**Table 1. Run-time boundaries for the workloads (seconds)**

turning it as a prediction. As has been introduced before, this model is substantially different form the f-model, since the interval for the predictions is  $[max(1, rt - \alpha * rt), rt + \alpha * rt]$ , while the f-model is  $[rt, \alpha * rt]$ . On the other hand we have evaluated the effect of adding this kind of errors to determined typologies of jobs:

1. *High number of processors*: Jobs that use more than  $p_{up}$ <sup>2</sup> processors.
2. *Low number of processors*: Jobs that use less than  $p_{lb}$  processors.
3. *Short jobs*: Jobs whom runtime is less than  $rt_{lb}$ .
4. *Large jobs*: Jobs whom runtime is bigger than  $rt_{up}$
5. *High area*: Jobs whom runtime multiplied by the number of processors that it use is bigger than  $area_{up}$
6. *Low area*: Jobs whom runtime multiplied by the number of processors that it use is lower than  $area_{lb}$
7. *More executed applications*: Jobs whom submitted application is one of the most executed in the hole workload.
8. *Low processors and low runtime*: Jobs whom runtime is less than  $rt_{lb}$  and whom number of used processors is less than  $p_{lb}$
9. *Not small jobs*: Jobs whom runtime is more than  $rt_{lb}$  or whom number of used processors is more than  $p_{lb}$
10. *Pure Equal*: All the jobs are prediction using the error and stdev provided as an input.

<sup>2</sup>The jobs are categorized based on a set of upper (*up*) and lower (*lb*) bounds that define which jobs matches to the descriptions. The computation of these bounds is introduced later.

11. *Equal*: A fixed percentage of jobs  $\partial$  are perfectly predicted. They are chosen using a random uniform variable. This means that there is no fixed criteria for select this subset of jobs. The rest of the jobs are predicted using a provided error and stdev.

The applications types' definitions are based on lower and upper bounds that are computed for each workload and variable (runtime, number of processors etc.). Therefore, the application types may differ from different centres, for example, the lower runtime bound for the LANL workload is 224.27 secs. while in the CTC is 1152.2 secs. Initially we defined these boundaries based on the statistical properties of the workload variables. For each of the workload variables (runtime and number of processors) we computed its percentiles and defined the lower bound as the 20th percentile of the variable and the upper bound as the 80th percentile. However, the experiments that used these bounds did not provide interesting results. It was caused due to the categories definitions were based in statistical properties rather than following a logical criteria. For example using the first criteria the definition of small jobs for the CTC center were all the jobs with runtime lower than 10 minutes, but jobs with runtime less than 1 hour were still small compared to the rest of the workload. For solve this problem we took similar approach that the once taken by Sui-Chiang et al. in [2]. We defined the boundaries following a reasonable criteria rather than statistical using as a basis for our decisions the percentiles for each variable for each workload. The upper bound has been defined as the  $ub_{\alpha} = 95_{th}Percentile/2$  and the lower bound as  $lb_{\alpha} = ub_{\alpha} * 0,05$ , where  $\alpha$  is the workload variable: runtime, number of processors etc.

The inputs for the experiments when evaluating the impact of the quantitative errors in the scheduling are:

1. The **policy** and **workload trace** used in the simulation (EASY-backfilling, SJB-First or LXF&W(w)-Backfilling).
2. The **error** and **standard deviation** used in the prediction generation. The deviation is used for not apply exactly the same error in all the predictions. Thus, we can also evaluate the effect of having high dispersion in the prediction errors.
3. The **type of jobs** that are predicted with high accuracy (introduced in the first in initial part of the section). When the predictor is asked for an prediction, if the job whom prediction is computed matches to this type, a perfect prediction will be returned (it will be the real runtime) otherwise the default error will be applied on the prediction.

The runtime prediction provided by the predictor for the job is computed as  $job_{RtPred} = job_{rt} + job_{rt} * U[0,0.05]$

(an almost perfect prediction) when it matches the category that is evaluated in the experiment. Otherwise, the returned prediction will follows the uniform distribution  $U[\max(1, job_{rt} - \frac{\alpha * job_{rt}}{100}), job_{rt} + \frac{\alpha * job_{rt}}{100}]$  where the  $\alpha$  follows the normal distribution  $\alpha \sim N(error, stdev)$ .

In the equal category approximately a 50% of the jobs are predicted with accuracy. In the first approach we applied the error to the 100% of jobs (pure equal category), however later on we found this approach quite pessimistic. We decided to apply the error to a 50% percent of the jobs (equal category). Our approach was based on two different ideas. The first one is presented in [2], where is concluded that with a 60% of the jobs provide approximately accurate requested runtimes the scheduling policy performance was improved. The second one, was that we saw that when using the other categories approximately a 30-55% of the jobs where predicted with accuracy.

## 5.2 Prediction Generation Using Qualitative Errors

In the above subsection we have presented how the quantitative errors are computed. As will be presented in the *experiments evaluation* section, once the experiments with these kind of errors were analyzed, we realized that although adding an amount of time in the prediction has a clear effects in some categories (mainly to the short jobs) we could be more aggressive. We decided to test what would occur with the system performance if with the job prediction the nature of the job was changed, for example predicting that a job is short while it is large. This section provides the descriptions of the experiments that we designed for test the impact of such kind of errors.

The qualitative errors model behaviors that predictors can frequently have while users should not: completely mistake the job runtime prediction. Usually, users tend to overestimate their jobs around two or even four times the real runtime of the job. This is caused because they have an approximate idea about which is the required time for their applications and they want to avoid their jobs to be killed. However, predictors probably will not have the same knowledge concerning the submitted applications and they could make huge mistakes in the job runtime predictions. The experiments application categories that have been evaluated for this qualitative errors are:

1. *short2large*: A percentage of  $\phi$  short jobs that will be predicted as large jobs predicting its runtime bigger than  $rt_{ub}$ .
2. *large2short*: A percentage of  $\rho$  large jobs that will be predicted as short jobs predicting its runtime lower than  $rt_{lb}$ .

3. **short2large** and **large2short**: A percentage  $\omega$  of large and short jobs that will be predicted as short jobs and large jobs respectively.

The inputs for the experiments when evaluating the impact of the qualitative errors in the scheduling are:

- The **policy** and **workload** used in the simulation (we used the same policies that in the quantitative analysis).
- The **percentage** and **standard deviation** of jobs to which prediction will imply changing their nature.
- The **conversion** type that indicates which types of jobs are predicted with a runtime that changes its nature.

Given a random variable  $\gamma \in U[0..100]$ , the runtime prediction provided by the predictor when modeling the qualitative errors for the job is computed as  $job_{rt} + job_{rt} * U[0,0.05]$  if  $\gamma > \beta$  where  $\beta$  follows a the normal distribution  $\beta \sim N(\text{percentage}, \text{stdev})$ . Otherwise:

- If the job is a small job and the category that is being evaluated is "**short2large**" or "**short2large and large2short**" then the prediction is computed by adding the  $rt_{upperbound}$  to the real job runtime. Thereby, the prediction will indicate to the scheduler that the job is large while it is short.
- If the job is big and the category evaluated is "**large2short**" or "**short2large and large2short**" the prediction is done by subtracting the  $rt_{upperbound}$ . In the case that the job prediction is bigger than the lower bound (can happened with the largest jobs) it is returned with the static value  $rt_{lowerbound} - 1$ .
- If the two previous conditions are not satisfied then initial almost perfect prediction is returned.

## 6 Experiment Characterization

Our study has been focused in a set of four different backfilling scheduling variants: EASY-backfilling, SJB-First and LXF&W(w)-Backfilling. We have used four different workloads: the San-Diego Supercomputer Center 95/96 traces (SDSC), the Cornell Theory Center (CTC) SP2 log, the Swedish Royal Institute of Technology (KTH) SP2 log, and the Los Alamos National Lab (LANL) CM-5 log.

The inputs for the simulations are basically: parameters that allows to choose which policy has to be used, which kind of prediction has to be used, parameters for tune the synthetic predictions used in the study and, finally, the workload trace that that will be used in the simulation. The trace is based in the standard workload format (SWF) proposed by Feitelson et al. [1] [5]

The experiments have consisted on simulating all the different backfilling policies but, instead of using the user estimation provided in the workload, we have provided to the scheduler the prediction generated in a fake predictor. Next two subsections present the experiments that have been carried out for the evaluation.

### 6.1 Quantitative Errors Experiments

The parameters for the experiments carried out for evaluate the impact of quantitative errors in the runtime prediction are presented below:

- **Policies:** EASY-Backfilling, SJB-First and LXF&W(w)-Backfilling.
- **Errors:** the following errors have been used: {5, 100, 200, 400, 600, 700, 800, 1000, 10000}.
- **Standard deviations:** joint to the last errors two groups of standard deviations have been used: {0.5, 2, 10, 15, 20, 25, 25, 25, 25} and {0.5, 20, 40, 60, 80, 120, 200, 300,1000}.
- **Quantitative Categories:** 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

We tested two different standard deviations due to we want to experiment which is the effect of adding more variation to the prediction (more chaos).

#### 6.1.1 Qualitative Errors Experiments

The parameters for the experiments carried out for evaluate the impact of qualitative errors in the runtime prediction are presented below:

- **Policies:** EASY-Backfilling, SJB-First and LXF&W(w)-Backfilling.
- **Percentage of jobs to be converted:** for those policies that use predictions the following percentages have been used {5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.
- **Standard deviations:** joint the last percentages the two groups of stdevs have been used {0.05, 0.1, 0.2, 1, 2, 3, 4, 4, 4, 4, 5, 5}.
- **Qualitative Categories:** 1, 2 and 3.

## 7 Experiment Evaluation

In this section the more relevant conclusions resulted obtained from the experiments presented in the previous section are presented. For lack of space we only present a subset of all the data that we have generated in the experiments.

Three different types of figures are discussed: first the performance for the pure equal category; second a comparison for the quantitative categories, including: the equal, short and large category; and finally the analysis of the qualitative categories is presented. The evaluation results for the quantitative prediction categories high/low number of processors, high/low area, more executed applications and low processors and low runtime have not shown relevant impacts on the scheduling performance. Thereby, their performance is not shown in the figure analysis.

## 7.1 Quantitative Prediction Errors

In general, we found out that there is no common pattern when adding quantitative errors on the job runtime prediction. The unique category that has shown an impact the performance of the scheduling policy is the large and the equal categories. However, in this last category when the quantitative errors are really high. Thus, the prediction accuracy for the short jobs is something important and that can let to good scheduling. We expected that prediction errors in the categories based on the amount of processors (high/low number of processors used and high/low area categories) would have higher impacts on the performance of the system. However, we have stated that carrying out accurate prediction on these topologies of jobs does not have dramatic effects.

### 7.1.1 Pure Equal Category

The first interesting fact that can be observed in the figure 1 is that the bounded slowdown for the pure equal shows a chaotic behavior and it does not follow any pattern. The bounded slowdown tendency for the workload `kth_sp2` shows higher values respect to the other two four workloads. This is caused due to a 40% of the jobs of this workload are really small (less than one minute), while in the rest of the workloads this percentage goes from 10% till a 20%. This clearly shows how the fact that a given workload has an important amount of really short jobs has an incredible effect in the global bounded slowdown when the job runtime predictions are inaccurate.

From figure 1, we can state there are no clear effects of the quantitative errors when they are applied to all the jobs in the bounded slowdown. Although using an error of 10000% the scheduling performance is not influenced. For example in the `sdsc_par96` figure with the SJF and LXWF policies and an error of 5% there is an bounded slowdown of 2 and 2,5 respectively, and there is an bounded slowdown of 1 and 1,4 for the same workloads and policies with an error of 1000%. This behaviour is caused because the pure equal category it is adding quantitative errors to all the job prediction. Using this category the global effect is that the

job runtime is over scaled and it has no effect in the backfilling. So adding a constant error in all the prediction has no collateral effects.

### 7.1.2 Large, Short and Equal Categories

In equal category, not only the performance of the system is not damaged by the inaccuracy of the prediction errors, in some situations it experiments an improvement of two one order of magnitude respect a perfect prediction. For example, the bounded slowdown for SJF Backfilling policy in the `l_kth_sp2` is reduced from 3 with an almost perfect prediction until 2 with an error on 1000%. The inaccuracy of the runtime estimates can let to a very good scheduling, and errors around 200% or even 400% in the runtime prediction can let to better schedule than perfect estimations. The impact of this inaccuracy has been named by Tsafirir as the Heel and toe effect in his studies about the impact of user runtime estimates in the backfilling policies [21]. However the scheduling performance starts to be affected by really high quantitative errors (more than 5000%)

The large category has demonstrated to have more effect in the performance of the system. In the figure 3 the bounded slowdown for the `ctc_sp2` workload and for all the policies presents a clear effect of the error when the large category is used. The large policy predicts with high accuracy large jobs, what means that the errors are mainly being added to the short jobs. Thereby, this study corroborates that the prediction accuracy (in this work positive and negative errors) on short jobs has relevant impact on the system performance. On the other hand, the short category, that predicts with accuracy short jobs and adds errors to the rest of jobs, is not highly affected by the increment prediction error. The short category shows that there are no clear effects of the predictions error in the non shorts jobs. The `kth_sp2`, `sdsc_par96`, and `lanl_cm5` workloads present similar patterns as the `ctc2_sp2` workload (see figures 2 4 and 5).

Similar to the conclusions that the user runtime estimates impact works achieved, we can state that accurate prediction of the short jobs is something important, independently of positive or negative errors. The accuracy for the large jobs, it is also important, however as their impact in the policy performance is not as dramatic as with the short jobs, higher errors are acceptable.

## 7.2 Qualitative Prediction Errors

The main goal of our study was to find out which kind of errors have real impact on the scheduling performance. Because we realized that there were clear effects with the quantitative errors in the categories that were defined based on the runtime rather than other variables, such as the num-

ber of processors, we decided to take more drastic measures: instead of adding quantitative errors in the categories short and large, we would change the nature of the jobs prediction by adding qualitative errors to the job prediction.

This new approach provided us also significant results, and corroborated the results obtained in the previous experiments. Figures 6, 7, 8 and 9 present the bounded slowdown for the different workloads evaluated in these experiments. There is a clear effect when using the categories **short2large** and **large2short** on the scheduling when incrementing the percentage of jobs whose nature is being changed: the performance of the system at least is decreased by a factor of two. For instance, in the `L.ctc_sp2` (figure 6) the bounded slowdown goes from 2, predicting as a large jobs a 5% of the shorts jobs, until a 20 using a percentage of 80% of the same jobs. In this last example the bounded slowdown is incremented by 4 times.

However, a heavier impact occurs when changes are applied to all the short and large jobs (using the category **"short2large and large2short"**). Obviously, the number of jobs that are changed in this case is bigger than in the other two categories. However, only changing the 50% of jobs that belongs to this category there is an important effect in the performance of the system, in some cases increasing the bounded slowdown by 7 times. Furthermore, in the worst cases, where the 90% of large and short jobs are affected by the change, the bounded slowdown can be incremented by 10 times.

The qualitative errors in the job runtime prediction have been demonstrated to have relevant impact in the performance of the system. We have demonstrated that incrementing the percentage of jobs affected for this kind of prediction errors an important lost of performance can be appreciated in the backfilling scheduling policies. This study provides important information that should be taken into account when designing predictors that will be used in backfilling scheduling policies. These predictors should try to avoid qualitative errors in their predictions, for example as some have been presented in some works, providing confidence interval in the predictions that could advise that the prediction could have either quantitative or qualitative errors.

## 8 Conclusions and Future Work

The study of the impact of the user runtime estimates accuracy in the performance of the scheduling policies has been deeply analyzed in several works. Using user runtime estimation models, such as the f-model, researchers have evaluated how the runtime accuracy provided by the user at the job submission can affects the performance of the backfilling policies and its variants. However, these traditional estimation models can not applied to backfilling scheduling

policies that use runtime predictions rather than user estimates.

In this paper we have described and evaluated a set of f-model based prediction models that characterize the behavior that prediction techniques have shown in HPC centers. They have been designed for evaluate scheduling policies that use predictions rather than user estimates. We have formalized two different type of prediction error models: quantitative errors and qualitative errors. The first group consists on adding or subtracting a  $\alpha$  percentage to the real runtime of a job and returning it as a prediction. We have defined a set of variants of this model based on to which kind of jobs the quantitative error has to be applied (i.e: to the jobs with large runtime or jobs that use large number of processors). The second group consists on changing the nature of the job with the prediction, for instance predicting the job as short while it is really large.

We have provided an evaluation of the impact of these prediction models to a set of backfilling variants. The analysis have stated that that adding quantitative errors to the runtime of all the jobs of the workload has no clear impact on the performance of the system, even with prediction errors of 10000%. However, when the errors are applied randomly to the 50% of all the stream of jobs the scheduling performance is clearly affected by high quantitative errors (more than 5000%). Surprisingly, the negative errors did not result in important impact on the performance of the system as we had expected.

If the quantitative errors are applied to determined subset of jobs based on the required resources (lower/high number of processors, lower/high memory used or lower/high *processors \* runtime* area consumed) there is no relevant impact in the performance of the system. However, when this quantitative prediction errors have been applied to the prediction of the short jobs the performance of the average bounded bounded slowdown experiments a smoothly increment until errors of 400%, and from this point it experiments a exponential increment. Taking into account the results obtained in this study, we support that it is not necessary sense to design specialized predictors for determined job types based on their resource required for achieve good performance. However, predictors should have to be as much accurate as possible for those jobs that are likely to be short, and try to avoid high quantitative errors on them.

Clearly, qualitative errors have to be avoided when predicting the job runtime for the short and large jobs. These errors have clear exponential tendency on the average bounded slowdown for all the evaluated workloads when the number of jobs being miss categorized is incremented.

As a future work we plan to study the impact of prediction techniques in distributed scheduling architectures. Specifically, we plan to study the impact of prediction errors in the scheduling policy described in [7]. In these new sce-

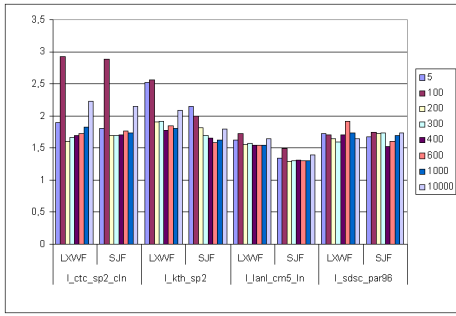


Figure 1. BSLD - Pure equal category.

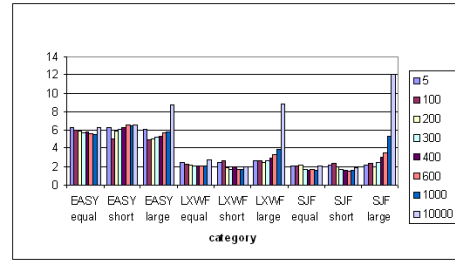


Figure 4. BSLD - Quant. Errors - KTH

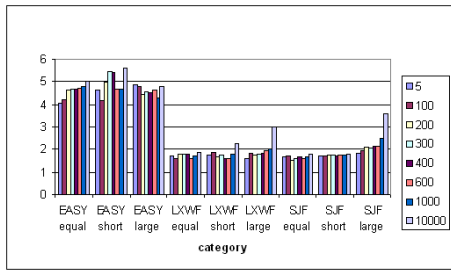


Figure 2. BSLD - Quant. Errors - SDSC

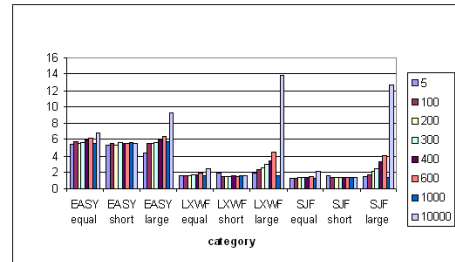


Figure 5. BSLD - Quant. Errors - LANL

nario we also plan to deploy specific prediction techniques based on datamining (for instance: decision trees or neural networks) and evaluate the impact of the amount of historical information in their prediction runtime accuracy.

## 9 Acknowledgments

This paper has been supported by the Spanish Ministry of Science and Education under contract TIN2004-07739-C02-01.

## References

- [1] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby.

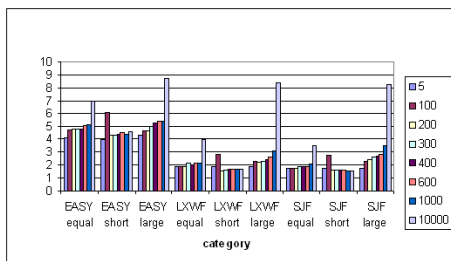


Figure 3. BSLD - Quant. Errors - CTC

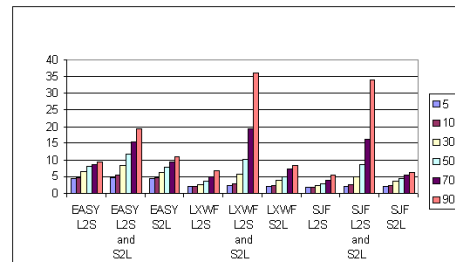


Figure 6. BSLD - Qual. Errors - CTC

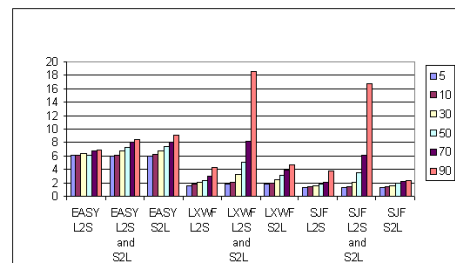


Figure 7. BSLD - Qual. Errors - LANL

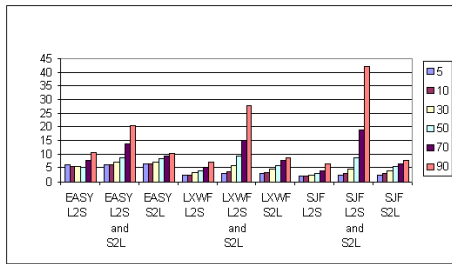


Figure 8. BSLD - Qual. Errors - KTH

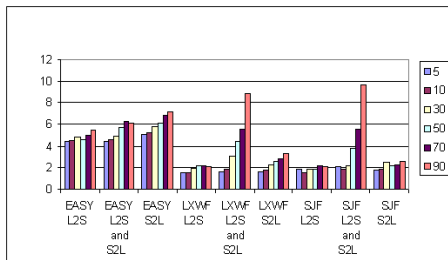


Figure 9. BSLD - Qual. Errors - SDSC

Benchmarks and standards for the evaluation of parallel job schedulers. *Job Scheduling Strategies for Parallel Processing*, vol 1659:pp. 66–89, 1999.

- [2] S.-H. Chiang, A. C. Arpaci-Dusseau, and M. K. Vernon. The impact of more accurate requested runtimes on production job scheduling performance. *8th International Workshop on Job Scheduling Strategies for Parallel Processing*, Vol. 2537:103 – 127, 2002.
- [3] M. V. Devarakonda and R. K. Iyer. Predictability of process resource usage : A measurement based study on unix. *IEEE Tans. Sofw. Eng.*, pages 15(12) and pp. 1579–1586, 1989.
- [4] P. Dinda. Online prediction of the running time of tasks. *Cluster Computing SIGMETRICS/Performance*, pages 225–236, 2002.
- [5] D. D. G. Feitelson. Parallel workload archive, 2006.
- [6] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling - a status report. *Job Scheduling Strategies for Parallel Processing: 10th International Workshop, JSSPP 2004*, 3277 / 2005:9, June 2004.
- [7] F. Guim and J. Corbalan. A job self-scheduling policy for hpc infrastructures. *Job Scheduling Strategies for Parallel Processing: 13th International Workshop, JSSPP 2007*, 2007.
- [8] R. Gibbons. A historical application profiler for use by parallel schedulers. *Job Scheduling Strategies for Parallel Processing 1997*, 1997.
- [9] A. Goyenechea, F. Guim, I. Rodero, G. Terstyansky, and J. Corbalan. Extracting performance hints for grid users using data mining techniques: a case study in the ngs. *Submitted to "Mediterranean Journal: Special issue on data mining*, 2006.
- [10] F. Guim, J. Corbalan, and J. Labarta. Analyzing loadleveler historical information for performance prediction. *Jornadas de Paralelismo 2005 and Granada*, 2005.
- [11] F. Guim, A. Goyeneche, J. Corbalan, J. Labarta, and G. Terstyansky. Grid computing performance prediction based in historical information. *Integrated Research in Grid Computing. November 2005. CoreGRID Integration Workshop*, 2005.
- [12] F. Guim, A. Goyeneche, I. Rodero, and J. Corbalan. The grid backfilling: a multi-site scheduling architecture with data mining prediction techniques. *Second CoreGRID Workshop on Middleware at ISC2007*, 2007.
- [13] D. Kerbyson, J. Harper, A. Craig, and G. Nudd. Pace: a toolset to investigate and predict performance in parallel systems. *Proceedings of the of the European Parallel Tools Meeting*, page 6, October 1996.
- [14] K. Kurowski, A. Oleksiak, J. Nabrzyski, A. Kwiecien, M. Wojtkiewicz, M. Dyvzkowski, F. Guim, J. Corbalan, and J. Labarta. Multi-criteria grid resource management using performance prediction techniques. *Integrated Research in Grid Computing. November 2005. CoreGRID Integration Workshop*, 2005.
- [15] H. Li, J. Chen, Y. Tao, D. Groep, , and L. Wolters. Improving a local learning technique for queue wait time predictions. *Cluster and Grid computing*, 2006.
- [16] J. Skovira, W. Chan, H. Zhou, and D. A. Lifka. The easy - loadleveler api project. *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes In Computer Science; Vol. 1162 archive:41 – 47, 1996.
- [17] W. Smith. Improving resource selection and scheduling using predictions. *Grid resource management: state of the art and future trends*, pages 237 – 253, 2004.
- [18] D. Talby and D. Feitelson. Supporting priorities and improving utilization of the ibm sp scheduler using slack-based backfilling. *Parallel Processing Symposium*, pages pp. 513–517, 1999.
- [19] D. Talby, D. Tsafirir, Z. Goldberg, , and D. G. Feitelson. Session-based and estimation-less and and information-less runtime prediction algorithms for parallel and grid job scheduling. Technical report, School of Computer Science and Engineering and The Hebrew University of Jerusalem, 2006.
- [20] D. Tsafirir, Y. Etsion, , and D. G. Feitelson. Modeling user runtime estimates. *In the 11th Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Vol.3834:pp. 1–35, 2006.
- [21] D. Tsafirir, Y. Etsion, and D. G. Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. *In the IEEE TPDS*, 2006.
- [22] Y. Zhang, W. Sun, , and Y. Inoguchi. Cpu load predictions on the computational grid. *Cluster and Grid computing*, 2006.