

Impact of Qualitative and Quantitative errors of the job runtime estimation in backfilling based scheduling policies

F. Guim, J. Corbalan, J. Labarta¹
Barcelona Supercomputing Center
{francesc.guim,julita.corbalan,jesus.labarta}@bsc.edu

Abstract

Estimation or prediction accuracy in backfilling based policies it's an important issue that has high impact on the system performance. However it's not clear which is the required precision of such estimations, moreover it's not clear which kind of errors are critical when scheduling the jobs that are queued in the local systems. In this paper we present a deeper analysis of the impact of the estimation errors in the scheduling. The study is based on several criteria. For instance the implications of carrying out accurate predictions of determined kind of jobs, having qualitative errors or quantitative errors and so on.

As far as we know, all the works that have analyzed the impact of the estimation error in the backfilling policies based they results and conclusions in a set of well known metrics, such as the average slowdown or the wait time. We present a new parametrical metric that evaluates the index of satisfaction of the user. As shown in the paper, conclusions obtained when evaluating backfilling based policies with this metric may differ with other conclusions obtained, for example, with average slowdown.

1 Introduction

Backfilling based scheduling policies have been demonstrated to be one of the scheduling policies that achieve higher performance in parallel architectures. However one the major problem that the core of algorithm has is that the runtime of the scheduled applications is supposed to be known, or at least a closer estimation at submission time.

Many works can be found in the literature concerning evaluations of backfilling policies variants and their performance. For instance, the research group of D. Feitelson et. al have presented several analysis of the EASY backfilling and its variants [6]. Furthermore many works have presented the impact of user estimates in such policies, such as the once presented by Tsafirir et al. in [2]. However, all these studies are mainly focused on the impact of quantitative errors in the performance of the system, for example presenting the impact of

an error of 50% in the user runtime estimation. Moreover, as far as we know in the literature has not been presented the impact of errors in prediction in an specific kind of jobs. For example studying which are the effects of carrying out accurate predictions of those jobs that, compared to the rest of submitted jobs, use less number of processors.

In this paper we present a detailed study of which is the impact of the error estimation in the scheduling. We evaluate two different types of errors: quantitative errors, as mentioned before, adding an amount of time in the real runtime; and qualitative errors, changing completely the nature of the job, for example estimating that a job will be really large while it is short. Furthermore, we have evaluated the impact of predicting with accuracy a given subset of jobs based on their nature, for example jobs that use a reduced number of processors or jobs that use a high amount of memory. The main goal of this second study have been detecting if predicting with accuracy a given kind of applications the performance of backfilling increases substantially. In affirmative case would make sense to design specialized predictors for these type of applications.

A crucial factor, when studying the impact of a given policy in the performance of the system, is the metric used in the evaluation. In most of the works there is a set of well-known and accepted metrics that are used for the mentioned evaluation. In general the most used metrics are: the average slowdown and the average wait time. However, in some works, as a measure of utilization of the system, the throughput is also used.

In our opinion the average slowdown is a good metric since provides a normalized scale about how much a user has to wait for a job completion. For example, in the case that a job has to wait one hour but only uses one second of processor will have a high slowdown, meaning that compared to its runtime it has waited an unacceptable amount time; but a job that also waits one hour but uses one hour of processor will have a slowdown of 2, what can be more reasonable from the point of view of a user. The main problem of this metric is that is highly biased by the response time of those jobs that have less runtime, and in general short jobs tend to have really high slowdowns. It's a metric that is clearly focused on the optimization on the wait time for this kind of jobs. On the other hand, for large jobs we found the difficulty of distinguishing the really impact of a given policy when using the average slowdown as a response performance variable. For instance a

¹This paper has been supported by the Spanish Ministry of Science and Education under contract TIN2004-07739-C02-01.

slowdown of 1.5 may be acceptable for a job that execution takes 1 hour, but it's not clear if the same conclusion can be obtained for a job that takes 1 week.

The other most used metric, the average wait time, does not provide a clear description of how well the system is scheduling the jobs, for example conclusions can be biased by large jobs that have large wait time. In some cases, more concise conclusions can be obtained by computing the wait time using intervals. For instance, where each interval is a bucket that contains the wait of a given type of jobs (f.i: short jobs, medium jobs and large jobs). However, in this case we would have three metrics rather one, and the analysis of experiments can become in exponential difficulty. In general, optimizing this metric would mean improving the wait time for the large jobs, due to this kind of jobs are have most incidence to such metric.

In our opinion the main lack of the two mentioned metrics, and derivatives, is that they do not provide a unified way for evaluate a scheduling policy due to they give different importance to the jobs, as mentioned one is more influenced by the small jobs and the other by large jobs. In this paper we present a new metric that provides a uniform index of satisfaction of the user about the submitted jobs. Our main goal have been designing a metric that tries to evaluate the Quality of Service that the user perceives, providing a uniformed way to evaluate all submitted jobs. It is not intended to be substitute of the other used metrics, rather this; it is intended to provide complementary information that we consider that it is not present in the other.

The rest of the paper is organized as follows: next section presents the related work; section 3 presents the characterization of the experiments that we carried out; section 4 presents the definition of the new metric used for the evaluation; section 5 provides the discussion about the impact of the qualitative errors versus quantitative errors; and finally the last section provides the conclusions of the presented work and the future work.

2 Related work

Our study has been focused in a set of four different backfilling scheduling variants: EASY-backfilling, SJF-Backfilling, FCFS-Backfilling and LXF&W(w)-Backfilling; and the simplest scheduling algorithm FCFS. Several works about these policies and schedulers that implement them can be found in the literature. In [13] Skovira et. al presented the first paper about the EASY algorithm and its performance in the LoadLeveler system. General descriptions about the most used backfilling variants and parallel scheduling policies can be found in the report that Dror. G. Feitelson et al. provides in [6]. Moreover, deeper description of the conservative backfilling algorithm can be found in [1], where the authors present it characterization and how the priorities that can be used when

choosing the appropriate job to be scheduled.

More analytical works are also present in the literature. Ahuva and Feitelson provide a comparison between the more conservative backfilling approaches with the aggressive version in [5]. They also get an interesting conclusion that is that backfilling actually works better when users tend to overestimate the runtime by a substantial factor. In [19] Sui-Hui Chiang et al. examine how the accurate requested runtimes can improve the system performance in high performance backfill policies. They examine the effect of the estimations errors in the backfilling derived policies FCFS, SJF, LXF and LXF&W(w), and conclude that in the heavier system loads there is a substantial improvement having accurate estimations.

Tsafir et al. carried out a similar analysis of the backfilling variants and the impact of error estimations, but using runtime predictions in instead of using user estimates in [2]. Regarding prediction, there are other works that have proposed several predictions techniques and how they can be applied in scheduling policies [22][4][21][16][3], however it is still an open problem.

The experimental part of our work [10] is also closely related with the workload analysis [17][15]. The conclusions of our experiments have been contrasted and based with the analytical studies available for each of the workloads that we have used in our simulations:

- The San-Diego Supercomputer Center (SDSC) Paragon logs from 1995. In [14] can be found a comparasion between the Paragon '95 workload with the NASA Ames iPSC workload.
- The San-Diego Supercomputer Center (SDSC) Paragon logs from 1996.
- The Cornell Theory Center (CTC) SP2 log [9].
- The Swedish Royal Institute of Technology (KTH) SP2 log.
- The Los Alamos National Lab (LANL) CM-5 log

Initially we also used workloads synthetically generated that were based on models such as [4][12][15][8]. However, as the number of experiments to generate was considerably big we decided only to test our ideas with the mentioned 5 workloads.

3 Experiment characterization

3.1 Simulation framework

All the experiments have been conducted using a C++ event-driven simulator that was implemented and used by Tsafir et al. in the paper [2].

This modular simulator allows to implement several policies due to the core of the simulator that provides the event scheduling is independent to them. It also allows adding predictors modules. Therefore a given policy can easily use predictions for plan the schedule of the jobs.

The inputs for the simulations are basically divided in: a set of parameters that allows to choose which policy has to be used, which kind of estimation has to be used (in case that the policy requires one) and other parameters that we added for tune the synthetic estimations used in the study; and a workload trace that is based in the standard workload format (SWF) proposed by Feitelson et al. [18] [7].

The SWL traces contain a header where a description about the system where the workload was created, such as the number of processors in the system, number of queues, number of nodes and so on. The header is followed by all the job entries. Where each entry has information concerning the job life cycle. Table 1 provides some of the more relevant fields of an entry.

For the experiments we have used the cleaned versions of the workloads (they have been enumerated in the above section). These cleaned versions exclude some of the jobs that are considered to be abnormal executions to omit non-representative activity, such as the workload flurries [20]: high activity for individual users. For example the workload of San-Diego Supercomputer Center Paragon log has 5 different flurries.

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 a synthetic estimation generated in a fake estimator. Next two subsections present how this fake estimator computes the estimation depending on several experiment parameters. We call it **'fake'** due to its estimations are computed by adding or subtracting some value to the real runtime of the job taken from the workload.

3.2 Estimation generation using quantitative errors

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

1. The **Policy** used in the simulation. Basically it can be EASY-backfilling, SJF-Scheduling or LXF&W(w)-Backfilling.
2. The **error** used in the new estimation.
3. The **standard deviation** used when computing the applied error. We added the stdev due to we did not want to use always exactly the same error. Then we could also evaluate the effect of having high dispersion in the estimations errors.
4. The **type of jobs** that are estimated with high accuracy. When fake estimator is asked for an estimation, if the job whom estimation is being computed matches to this

Field	Description
Submit Time	The earliest time the log refers to is zero, and is the submittal time the of the first job
Wait Time	The difference between the job's submit time and the time at which it actually began to run.
Run Time	The wall clock time the job was running.
Number of Allocated Processors	In most cases this is also the number of processors the job uses.
Requested Time	This can be either runtime or average CPU time per processor.
Status	1 if the job was completed, 0 if it failed, and 5 if cancelled.
User information	User id and group id.
Executable	a natural number, between one and the number of different applications appearing in the workload.

Table 1: Standard Workload Format

type, a perfect estimation will be returned (it will be the real runtime) otherwise the default error (input 2) will be applied on the estimation. Table 2 enumerates the different type of jobs that have been defined.

5. The **workload trace** used in the simulation. As will be explained in "Defining the upper and lower bounds" the characterization of type of jobs provided as a input 4 are based on the workload that is being simulated.

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 form different centres, for example, the lower runtime bound for the LANL workload is 224.27 while in the CTC is 1152.2. The remaining part of the subsection provides an explanation of how the mentioned limits (upper and lower bounds) have been computed, how the quantitative errors are generated and how the experiments for their impact on the scheduling have been analyzed.

3.2.1 Defining the upper and lower bounds: Initially we defined these boundaries based on statistical properties of each workload. For each of the variables we computed its percentiles [26] and define the lower bound as the 20th percentile of the variable and the upper bound as the 80th percentile. However, the experiments done with this configuration did not provide interesting and clear results. We found that it was caused due to the categories definitions were based in statistical properties. For example the definition of small jobs could be all the jobs with runtime was less than 10 minutes, due to a 20% of the jobs had less than 10 minutes, but a jobs with runtime less than 1 hour were still small compared

Category	Description
High number of processors (1)	Jobs that use more than $p_{upperbound}$ processors.
Low number of processors (2)	Jobs that use less than $p_{lowerbound}$ processors.
Short jobs (3)	Jobs whom runtime is less than $rt_{lowerbound}$.
Large jobs (4)	Jobs whom runtime is bigger than $rt_{upperbound}$.
High area (5)	Jobs whom runtime multiplied by the number of processors that it use is bigger than $area_{upperbound}$.
Low area (6)	Jobs whom runtime multiplied by the number of processors that it use is lower than $area_{lowerbound}$.
More executed applications (7)	Jobs whom submitted application is one of the most executed in the hole workload.
Low processors and low runtime (8)	Jobs whom runtime is less than $rt_{lowerbound}$ and whom number of used processors is less than $p_{lowerbound}$.
Not small jobs (9)	Jobs whom runtime is more than $rt_{lowerbound}$ or whom number of used processors is more than $p_{lowerbound}$.
Pure Equal (10)	All the jobs are estimated using the error and stdev provided as an input.
Equal (11)	A fixed percentage of jobs ∂ are perfectly estimated. 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 estimated using the provided error and stdev.

Table 2: Application categories for quantitative errors

Run-time bound	Applied formula
Upper bound	$50\% (95^{th} percentile(runtime))$
Lower bound	$5\% (rt_{upperbound})$

Table 3: Run-time boundaries definition

Workload	RT boundaries ¹		%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 4: Run-time boundaries for the workloads

to the rest of the workload. For solve this problem we took similar approach that the once taken by Sui-Chiang et al. in [19]. We defined the boundaries following a reasonable criteria rather than statistical, however using as a basis for our decisions the percentiles for each variable for each workload. Table 3 presents the description of how the lower and upper bounds for the run-time are computed. As an example, Table 4 shows the upper and lower bounds computed for each of the workloads. All the boundaries for each variable and workload can be found in [11]. The bounds for the variable *number of processors* is computed in a similar way.

3.2.2 Computing the quantitative error : The runtime estimation provided by the fake predictor is computed as shown in below. However, some remarks have to be presented concerning it:

- When the same estimation errors are applied to all the jobs (equal category), approximately a 50% of the jobs are estimate with accuracy. Initially we applied the error to the 100% of jobs, however we found this approach quite pessimistic. We decided to apply the error to a 50% percent of the jobs. Our approach was based on two different ideas. The first one is presented in [19], where is concluded that with a 60% of the jobs provide approximately accurate requested runtimes the runtime 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.
- In [5] Feitelson et. al concluded that a over estimation in the runtime improved substantially the performance of the system. We decided to make it more real subtracting in a 50% of the estimations the error rather than adding it. Usually user won't under estimate the runtime, due to in backfilling policies these jobs are killed.

However in cases where a predictor is used it can make predictions that are lower than the real runtime.

The following source code presents the estimation computation.

```

input: JOB job, enum category
      double stdev, double error,
      int workload
output: double estimation

NormalDistribution jobEr(error, stdev);
UniformDistribution uniform(0,100);
double E = 0;

//Computing the error
if(category = equal &
    uniform.generateRandom() > 50)
    e = jobEr.generateRandom();
else if(matches(job,category,workload))
    e = 0;
else
    e = jobEr.generateRandom();

//Generating estimation
if(uniform.generateRandom() > 50)
    estimation = rt + E*job.rt;
else
    estimation = |rt-E*job.rt|;

return estimation;

```

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

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

We tested two different standard deviations due to we want to experiment which is the effect of adding more variation to the estimations (more chaos). For some reason, as noticed in [6], it seems that giving more variations in the estimations to the scheduler when it implements a backfilling policy it achieve better performance. We decided that in some way this phenomenon had to be included in the experiments.

Category	Description
short2large (1)	The percentage (input 2) of short jobs that will be converted to large jobs. Overestimating it runtime until its bigger than $rt_{upperbound}$.
large2short (2)	The percentage (input 2) of large jobs that will be converted to short jobs. Underestimating it runtime until its lower than $rt_{lowerbound}$.
short2large and large2short (3)	The percentage (input 2) of large and short jobs that will be converted to short jobs and large jobs respectively.

Table 5: Application categories for qualitative errors

3.3 Estimation 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 such kind of error were analyzed, we realized although that adding an amount of time in the prediction has a clear effects in some categories (mainly in the large category) we could be more aggressive.

We decided to test what happened if with the job estimation the nature of the job was changed, for example estimating 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 inputs for the experiments when evaluating the impact of the qualitative errors in the scheduling are:

- **The policy** used in the simulation (we used the same policies that in the quantitative analysis).
- **The percentage** of jobs to which estimation will imply changing their nature.
- **The standard** deviation used in the joint the above percentage.
- **The conversion** type that indicates which types of jobs are estimated giving a runtime that changes its nature.
- **The workload** trace used in the simulation.

As mentioned before, in these experiments the parameter **conversion type** has a different meaning that in the previous once. Table 5 provides all the types that have been defined.

Like in the quantitative experiments, the presented conversions types are based in the lower and upper runtime time values. These values are computed in the same way as presented preceding subsection.

3.3.1 Computing the error: The following source code presents the general algorithm used for compute the estimation of the runtime that it's provided to the scheduler when required.

```

input: JOB job, enum conversion
      double stdev, double perc,
      int workload
output: double estimation

NormalDistribution jPerc(perc, stdev);
UniformDistribution uniform(0,100);
double E = 0;
double p = jPerc.generateRandom();
double unif = uniform.generateRandom();

if(job.rt < rt_lowerbound)
  if((conversion = 1 || conversion = 3)&
     p > unif)
    estimation = job.rt+rt_upperbound;
else if(job.rt > rt_upperbound)
{
  if((conversion = 2 || conversion = 3)&
     p > unif)
  {
    estimation = job.rt - rt_upperbound;
    if(estimation > rt_lowerbound)
      estimation = rt_lowerbound -1;
  }
}
return estimation;

```

If the job, of which estimation is required, is a small job and the uniform random value (0..100) is less than the percentage generated from the percentage normal distribution, and the conversion type is "**short2large**" or "**short2large and large2short**" then the result estimation is done by adding the $rt_{upperbound}$.

On the other hand, in the case that the job is big and the random value is less than the percentage generated from the normal distribution, and the conversion type is "**large2short**" or "**short2large and large2short**" estimation is done by subtracting the $rt_{upperbound}$. In the case that the job estimation still remains bigger than the lower bound (can happened with the largest jobs) the estimation is returned with the static value $rt_{lowerbound}-1$.

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

- **Policies:** FCFS, EASY-Backfilling, SJF-Backfilling and LXF&W(w)-Backfilling.
- **Percentage of jobs to be converted:** for those policies that use estimations 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 }.
- **Categories:** 1, 2, 3.

At this point we have presented all the experiments characterization: including the simulations inputs and paramters used in the study; and the estimation generation for the qualitative and quantitative errors used in them. In the following subsection we present the motivation for the usage the new metric that we have designed for the evaluation of all the simulation, its definition and peculiarities.

4 User-Quality of Service metric

In the introduction we presented several problems concerning the most commonly used metrics that are used for evaluating the systems performance.

The slowdown is highly biased by the short jobs, and does not provide how the large jobs are being affected by a given policy. Some researchers have used normalized slowdown metrics for avoid such effect. For example [2] use the average bounded slowdown as a performance metric, that is the slowdown for all the jobs, but those slowdowns that are bigger than a value σ are normalized to lower values. However, these approaches still have the problem that the effect of the policy in large jobs it's not clear.

On the other hand, the wait time is a metric that has been used for estimate how affected are the large jobs, but it does not provide a clear measure of how much they are punished. It is affected for many different job types, and can be also highly biased for large jobs that wait long time.

In this section we present the definition of a new metric that we designed for evaluate the performance of the system that tries to avoid the mentioned problems. We call it user **User-Quality of Service** (U-QoS). The general idea is to modelize the satisfaction of the user that has submitted the job. As has been shown in the earlier part of this paper, the level of satisfaction of the user higy depends on the nature of the submitted job. We expect that users that submit large jobs should be equally satisfied that users that submit short jobs.

In our opinion the user satisfaction for the submitted large jobs is well represented by its slowdown. This normalized value use to be between 1 and 2, and minor changes in this value use to be significant. For example if a large job improves it slowdown form 1.5 to 1.2 can be considered an important improvement in its response time. Thereby, for large jobs we use the slowdown computation as its U-QoS. On the other hand, the slowdown, as has been discussed before, it's not representative for short jobs. We focussed on finding out how the wait time for short jobs can be transformed to a value that has the same characteristics that the slowdown has for

the large jobs. The main idea of the User-Quality of Service is to normalize the wait time of this jobs using a value ϕ that depends on "how much" the large jobs have been waited in average and in what we call the **base wait time**. This wait time is a representative value for the short jobs that models what is a reasonable time that a short job should have to wait. For example the user can consider that a wait time of two minutes for a job that takes one minute to be executed is ok, and he/she can still consider that the same wait time for a job that takes 1 second to run is still ok, due to this amount of time from the user point perspective is reasonable.

Although the U-QoS mainly evaluates the equal satisfaction of short and large jobs, it can be parametrically set giving more weight either large or short jobs. This property may be desirable for determined types of centers where a given type of jobs must have more priority than others. For instance, in HPC centers large jobs can be prioritized compared to short ones. The formula 6 describes the computation of the metric.

There are some key concepts or some remarks concerning the above equation that have to be discussed:

- ϕ represents the normalization of the wait time respect the run-time of large jobs (wait/run).
- δ represents amount of wait time that is considered reasonable for a short job. It is important to see that:
 - It takes into account the amount of wait that the large jobs are having. It implies that the User-Quality of Service of short jobs is also dependant of how much satisfied are the large jobs.
 - The amount of time δ that is used for normalize the wait time of large job is computed by using the $rt_{lowerbound}$.
 - The impact of large jobs or small jobs in the global computation of the metric can be tuned using the parameter factor. Table 8 provides some of the values that we have used and which are the effect on the computation.
- For the rest of the jobs we have used the minimum of the lasts two values. Using the minimum we clean the noise that can be added by the intermediate jobs. Deeper analysis must be done in other to find out if there is any computation for this jobs that modelize with more accuracy their U-QoS.

In case that using the factor values in (2) and the scheduling policy priors large jobs the metric will have lower values for this jobs. On the other hand, if using the factor values used in (3) and the scheduling policy prioritize small jobs, the U-QoS will have lower values for this other kind of jobs.

$$\begin{aligned}
 & integerfactor[0..N] \\
 & \phi = \sum_{\forall jobs} \frac{wait_{job}}{run_{job}} \\
 & \delta = (factor \cdot \phi) \cdot rt_{lowerbound} \\
 \\
 & metric = \sum_{\forall jobs} \begin{cases} sld_{job} & job \in large \\ wait_{job}/\delta & job \in short \\ \min(sld_{job}, wait_{job}/\delta) & otherwise \end{cases}
 \end{aligned}$$

Table 6: User-Quality of Service computation

Factor value	Meaning
1 (1)	This value gives the same relevance to small and large jobs.
2,4 (2)	These values give more relevance to the large jobs than short jobs.
1/2,1/4 (3)	These values give more relevance to the short jobs than the large jobs.

Table 7: Factor values used in the experimentation

5 Experiment evaluation

In this section the more relevant data resulted from the experiments presented in the previous section are presented. For avoid presenting too much graphics and data in the paper we only present a subset of all the data that we have generated on the experiments. In [11] are available de rests of the data that may be useful or of interest to the reader.

5.1 Quantitative errors

In this subsection we present the more representative results that we obtained with the analysis of the impact of quantitative errors in the performance of the system. We present the results with the slowdown and the User-Quality of Service for each of the analyzed scheduling policies. Three different set of figures are shown: the first one presents the performance for the pure equal category; the second one presents a comparison for the quantitative categories, including: the equal, short and large category; and finally the analisis of the qualitative categories is presented. As have been explained in the above sections the main goals of this analysis was to evaluate the effect of:

- Adding quantitative errors to all the runtime estimation of the scheduled jobs (pure equal) .
- Adding quantitative errors to a specific subset of jobs: jobs that have an amount of runtime considerably big (large) and jobs that whom runtime is small (short).
- Adding quantitative errors to a random subset of the jobs (equal).

In general, we found out that there is a common pattern when adding quantitative errors on the job runtime estimation. Us-

ing an error of 100% in the runtime estimation, the performance on the system has a substantial improvement. This conclusion corroborates the works presented in [2] and [5]. On the other hand increasing the amount of error in the estimation does not have clear effects in the performance of the system. In general, there is nor negative neither positive tendency of the performance metrics with the exception of the large category.

5.1.1 Pure equal category: In the presented figures, for provide a more clear view, we have removed the EASY backfilling values due to they performance is substantially lower than the one presented by the other two backfilling policies.

The first interesting fact that can be observed in the figure 1 is that the slowdown for the pure equal shows a chaotic behaviour and it does not follow any pattern. The slowdown for the workload `kth_sp2` shows higher values respect to the rest. 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 slowdown. Comparing this slowdown with the metric presented in this paper, it can be observed that the U-QoS has more normalized values and it is more immune to the effect that the workload has an important amount of short jobs. This effect can be observed in all the rest of the figures presented in this section.

A surprising conclusion that can be obtained from figures 1 and 2 is that there is no clear effects of the quantitative errors in the slowdown neither in the U-QoS metric. Although using an error of 10000% no clear effects appear. For example in the `sdsc_par96` figure with the SJF and LXWF policies and an error of 5% there is an U-QoS of 1.6 and 1.7 respectively, and there is an U-QoS of 1.72 and 1.62 for the same workloads and policies with an error of 1000%. The maximum differences appear in the `ctc_sp2` workload, where the U-QoS goes from 1.89 and 1.8, with an error of 5%, to 2.2 and 2.4, with an error of 1000% in the SJF and LXWF policies.

Our impression is that this behaviour is caused due to the pure equal category it is adding quantitative errors to all the job estimation. 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 estimations has no collateral effects. We had expected that adding this error we would be adding more chaos to the system, and that would impact in the performance of the system.

At this point we decided check which was the effect of adding errors estimations in a subset of jobs. We though that this would have more impact in the system performance. We defined the categories presented in the section "Experiment characterization": the **equal category**, where the error is added to a 50% of all the jobs (rather than the 100% of the

jobs used in the **pure equal** category); the **short** category, where the jobs that are short were estimated perfectly and the error was applied to the rest of the jobs; and the **large** categories, where the large jobs were estimated perfectly and the error was applied to the rest of the jobs.

5.1.2 Large, short and equal categories: The large category is the one that has been demonstrated to have more effect in the performance of the system. In figures 3 and 4 the slowdown and the User-Quality of Service, for the `ctc_sp2` workload and for all the policies, present a clear effect of the error when the large category is used (This effect is more obvious in the U-QoS figure). The large policy estimates with high accuracy large jobs, what means that the errors are mainly being added to the short jobs. On the other hand, the short category, that estimates with accuracy short jobs and adds errors to the rest of jobs, is not highly affected by the increment of the applied error. In this second category the errors added to large jobs. Based on these two facts, we can conclude that accurate estimation of the short jobs is something important.

The accuracy for the large jobs, it is also important, however as the impact of the estimation of this kind of jobs is not as dramatic as with the short jobs, higher errors are acceptable. Figure 4 shows that the U-QoS metric presents a clear increment when the error is incremented, this behaviour does not appear with the same clarity in the slowdown figure. For this same workload the equal policy also presents a sort of ascendant pattern when the error is being increased. However the decrement of the performance is softer than in the large category, it starts to be significant with an error of 1000%. Probably it is caused due to the amount of short jobs that are being affected by the errors in this category is less that the amount affected in the smaller than in the large category.

The error has to be bigger for show some impact in both metrics. The `kth_sp2` presents similar patterns as the `ctc2_sp2` workload (see figures 5 and 6). The decrement of system performance can be also appreciated when the category used in the simulations is the large. Furthermore, the short category does not show any clear effect of the error in the estimation of the non shorts jobs. Again, the figure 6 shows that the U-QoS provides a different view and different information than the slowdown: high slowdown values are present in the figure 5. Mainly this is due to `kth` has an important amount of very short jobs and the global slowdown is affected by their slowdown values. On the other hand, in the U-QoS provides more normalized values, with similar scales (the slowdown for the `kth_sp2` grows from 40 of the SJF until 400 in the case of the EASY backfilling, while the U-QoS metric grows from 2 until 12). Moreover the values provided by our metric can be modelled easily than the slowdown.

We obtained similar conclusions for the workloads `sdsc_par96` when studying its slowdown and U-QoS graphics. Similar patterns can be observed in the presented figures.

5.2 Qualitative errors

The main goal of our study was to find out which kind of errors have real impact on the scheduling performance. At this point, seeing that there were clear effects with the quantitative errors in some categories, we decided to take more drastic measures: instead of adding quantitative errors, we would change the nature of the jobs adding qualitative errors to the job estimation.

This new approach provided us also significant results, and corroborated the results obtained in the previous experiments. Figures 12, 14, 16, 18 presents the U-QoS of 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 two times. For instance in `Lctc_sp2` figure (Figures 12) the UIS goes from 1.5, estimating as a large jobs a 5% of the shorts jobs, until a 6 using a percentage of 80% of the same jobs. In this last example the U-QoS 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 (what is similar to the number of jobs changed in the other two categories), there is an important effect in the performance of the system, in some cases increasing the U-QoS by 7 times. Furthermore, in the worst cases, where the 90% of large and short jobs are affected by the change, the U-QoS can be incremented by 10 times.

Using the slowdown as a performance metric, in general there are also similar patterns that show that increasing the percentage the slowdown is increased. For example, when evaluating the effect of the percentage in the category **short2large and large2short** for the `Llanl_cm5`, the slowdowns grows from 10, using an 5%, until 185, using an 90%, what means that the slowdown is incremented by 20 times.

As with the quantitative errors, the qualitative errors in the job runtime estimation have been demonstrated to have similar effects in the performance effects of the system. It's clear that depending on the goodness of the system this effects may be acceptable or not, however high percentage of jobs that are being affected for such kind of nature changes can imply an important lost of performance. 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. However, deeper study should be done for in the usage of real predictors in the backfilling based policies.

6 Conclusions and future work

In the paper we have presented the impact of quantitative errors and qualitative errors of the job runtime estimation on the performance of the backfilling based policies. We have also shown the effect of applying these errors to a determinate subset of jobs. The new performance metric User-Quality of Service has been presented. It has been demonstrated to be a useful and complementary metric to the existing metrics. The U-QoS use to remain stable and is less affected by extreme jobs, really short and really large jobs. In all the experiments it has demonstrated to be easier to understand and to have a behavior that can be easily modeled

In the experiments we have shown that that adding quantitative errors to all the runtime jobs estimation of the workload has no impact on the performance of the system. This is caused due to the final result of adding these errors are that the workload is only scaled but maintains the characteristics of a perfect estimation. Furthermore, in general, adding quantitative errors to a determined subset of jobs does not have 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. The errors in such kind of jobs have considerable effects in the presented performance metrics; adding quantitative errors in their job runtime estimation results in a clear increasing of the U-QoS and of the slowdown.

Qualitative errors in job runtime estimation have shown similar patterns as the other presents in the quantitative errors: adding qualitative errors to the short jobs has an important impact on the system performance. It has been demonstrated that adding qualitative errors on their runtime estimations rather results in dramatic drop of the performance.

We can conclude that the backfilling scheduling policies are being affected by the jobs runtime estimation error. However, not all the errors are critical, adding similar errors to all the jobs has no real impact, neither adding errors to those jobs that are large or medium. On the other hand, estimation errors of short jobs are more critical. If the estimations error increases, it results in a substantial increase of slowdown or U-QoS. Furthermore, qualitative errors in these jobs are more critical than the others, and users should try to avoid them.

Taking into account the results obtained in this study, we support that it make sense to design specialized predictors for determined job types, and use them instead of user runtime estimates as an input of backfilling scheduling policies. In general, this predictors should have to be as much accurate as possible for those jobs that are likely to be short, and try to avoid qualitative errors on them. On the other hand higher errors could be acceptable in the rest of the jobs.

Our current work is focused on evaluating the predictability of the backfilling policies. This study is intended to provide us some guidelines for designing scheduling policies for Grid

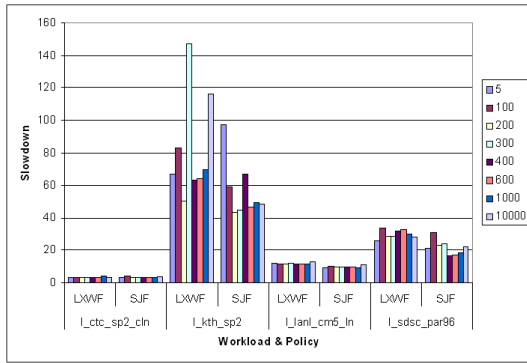


Figure 1: Slowdown using the pure equal category.

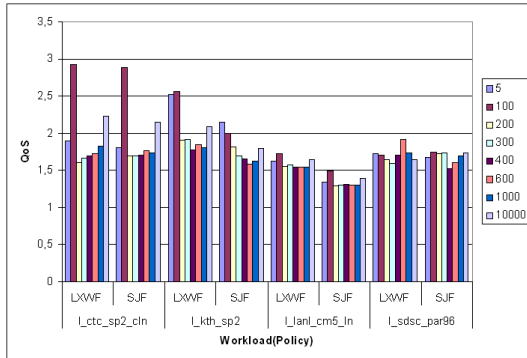


Figure 2: U-QoS using the pure equal category.

environments based on prediction techniques rather user runtime estimation.

References

- [1] D. F. D Talby. Supporting priorities and improving utilization of the ibm sp scheduler using slack-based backfilling. *Parallel Processing Symposium*, pages pp. 513–517, 1999.
- [2] Y. E. Dan Tsafir and D. G. Feitelson. Backfilling using runtime predictions rather than user estimates. *Submitted paper. Original Version is "Backfilling Using Runtime Predictions Rather Than User Estimates"*. *Technical Report 2005-*

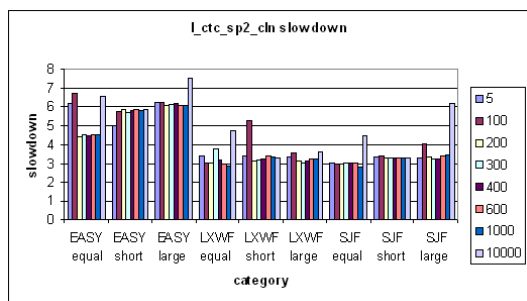


Figure 3: Slowdown using large, short and equal policies for the ctc_sp2

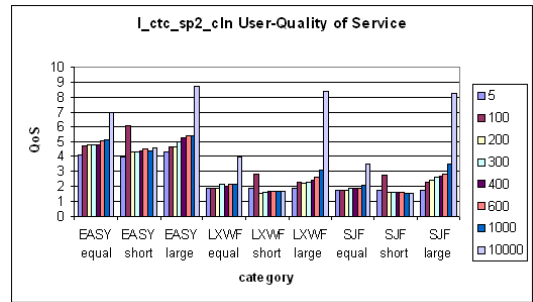


Figure 4: U-QoS using large, short and equal policies for the ctc_sp2

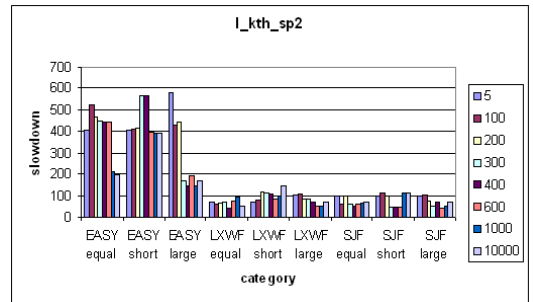


Figure 5: Slowdown using large, short and equal policies for the kth_sp2

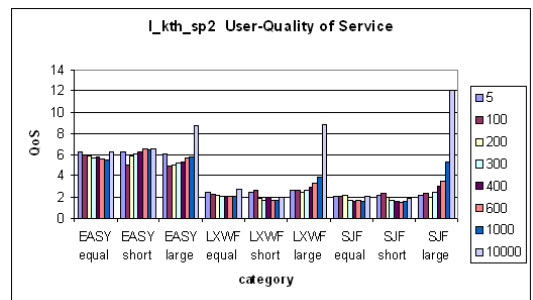


Figure 6: U-QoS using large, short and equal policies for the kth_sp2

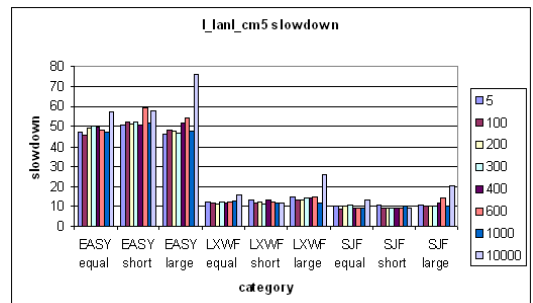


Figure 7: Slowdown using large, short and equal policies for the lanl_cm5

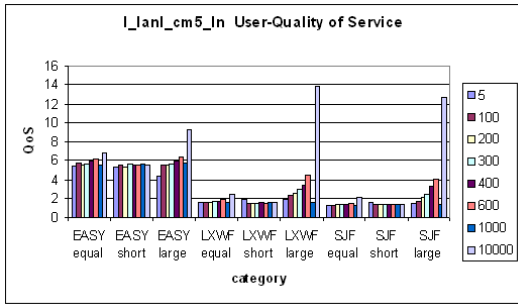


Figure 8: U-QoS using large, short and equal policies for the lanl_cm5

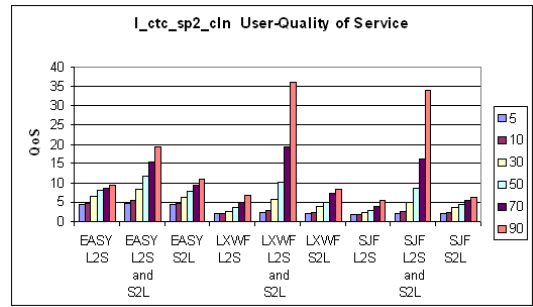


Figure 12: U-QoS using S2L, L2S and "L2S and S2L" policies for the ctc_sp2

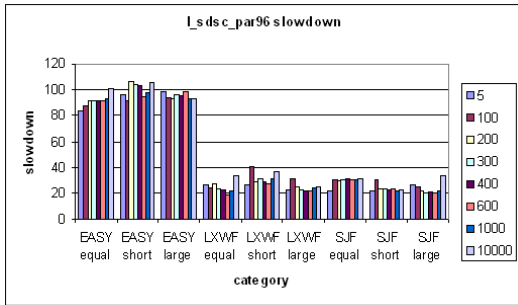


Figure 9: Slowdown using large, short and equal policies for the sdsc_par96

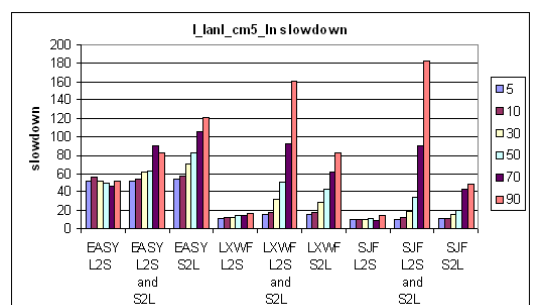


Figure 13: Slowdown using S2L, L2S and "L2S and S2L" policies for the lanl_cm5

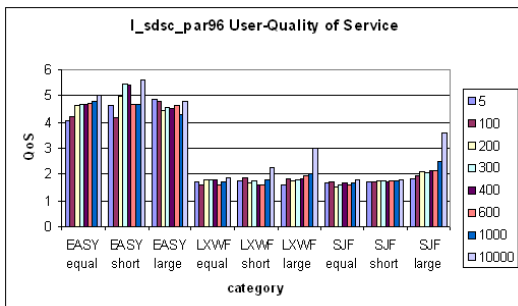


Figure 10: U-QoS using large, short and equal policies for the sdsc_par96

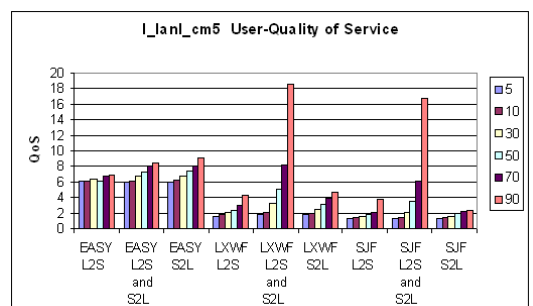


Figure 14: U-QoS using S2L, L2S and "L2S and S2L" policies for the lanl_cm5

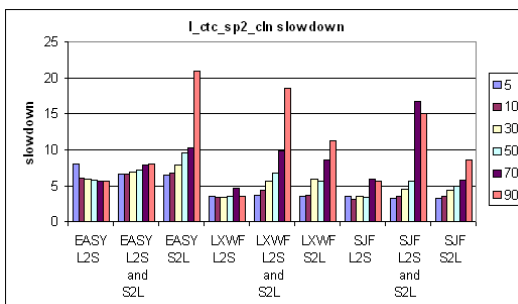


Figure 11: Slowdown using S2L, L2S and "L2S and S2L" policies for the ctc_sp2

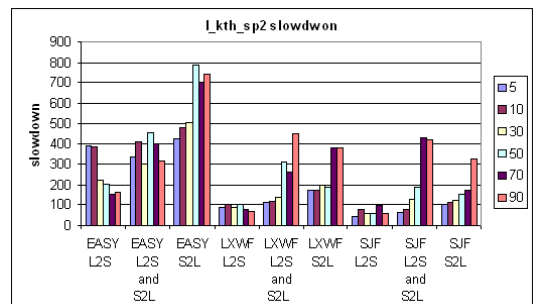


Figure 15: Slowdown using S2L, L2S and "L2S and S2L" policies for the kths_p2

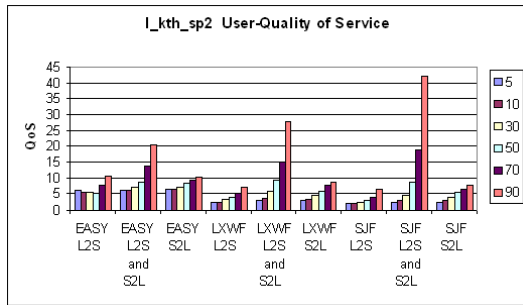


Figure 16: U-QoS using S2L, L2S and "L2S and S2L" policies for the kth.sp2

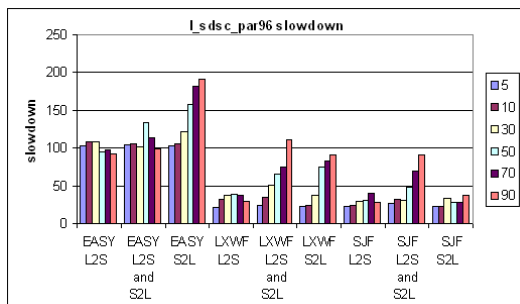


Figure 17: Slowdown using S2L, L2S and "L2S and S2L" policies for the sdsc.par96

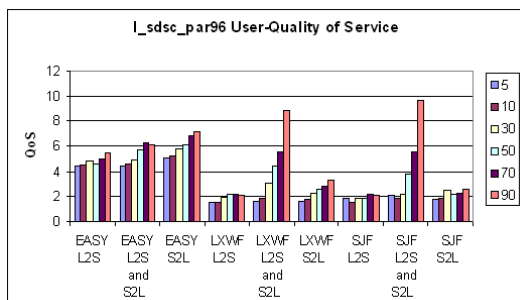


Figure 18: U-QoS using S2L, L2S and "L2S and S2L" policies for the sdsc.par96

5, School of Computer Science and Engineering, The Hebrew University of Jerusalem, pages pp. 513–517, 2005.

[3] P. Dinda. Online prediction of the running time of tasks. *Cluster Computing SIGMETRICS/Performance*, pages 225–236, 2002.

[4] A. B. Downey. Using queue time predictions for processor allocation. *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes In Computer Science; Vol. 1291:35 – 57, 1997.

[5] A. W. Dror G. Feitelson. Utilization and predictability in scheduling the ibm sp2 with backfilling. *Proceedings of the 12th. International Parallel Processing Symposium*, pages 542–546, 1998.

[6] L. R. Dror G. Feitelson 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] D. D. G. Feitelson. Parallel workload archive - <http://www.cs.huji.ac.il/labs/parallel/workload>, 2006.

[8] D. G. Feitelson. Packing schemes for gang scheduling?. *Job Scheduling Strategies for Parallel Processing*, Lect. Notes Comput. Sci. 1162:pp. 89–110, 1996.

[9] D. G. Feitelson and L. Rudolph. Workload evolution on the cornell theory center ibm sp2. *Job Scheduling Strategies for Parallel Processing*, 1162:pp. 27–40, 1996.

[10] J. L. Francesc Guim, Julita Corbalan. Analyzing loadleveler historical information for performance prediction. *Jornadas de Paralelismo 2005, Granada*, 2005.

[11] F. Guim. <http://francesc.guim.net/publications/quantvsqual> - data concerning the experiments done for this work., 2006.

[12] J. E. M. P. P. H. Franke, J. Jann and M. A. Jette. An evaluation of parallel job scheduling for ascii blue-pacific. *Supercomputing*, November 1999.

[13] H. Z. D. A. L. Joseph Skovira, Waiman Chan. 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.

[14] R. M. D. F. K. Windisch, V. Lo and B. Nitzberg. A comparison of workload traces from two production parallel machines. *6th Symp. Frontiers Massively Parallel Comput.*, pages pp.319–326, 1996.

[15] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *J. Parallel and Distributed Computing*, 11:pp. 1105–1122, November 2003.

[16] R. K. I. M. V. Devarakonda. Predictability of process resource usage : A measurement based study on unix. *IEEE Tans. Sofw. Eng.*, pages 15(12), pp. 1579–1586, 1989.

[17] G. S. Maria Calzarossa. Workload characterization: A survey. *Proc. IEEE*, 81:1136–1150, 1993.

[18] D. G. F. J. P. J. S. T. L. U. S. W. S. D. T. Steve J. Chapin, Walfredo Cirne. Benchmarks and standards for the evaluation

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

[19] M. K. V. Su-Hui Chiang, Andrea C. Arpaci-Dusseau. 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.

[20] D. Tsafir and D. G. Feitelson. Workload flurries. Technical report, School of Computer Science and Engineering, The Hebrew University of Jerusalem.

[21] I. T. F. Warren Smith, Valerie E. Taylor. Using run-time predictions to estimate queue wait times and improve scheduler performance. *Proceedings of the Job Scheduling Strategies for Parallel Processing*, Lecture Notes In Computer Science; Vol. 1659:202 – 219, 1999.

[22] V. E. T. Warren Smith, Ian T. Foster. Predicting application run times using historical information. *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes In Computer Science; Vol. 1459:122 – 142, 1998.