# Trace-Based Evaluation of Job Runtime and Queue Wait Time Predictions in Grids

Ozan Sonmez
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
o.o.sonmez@tudelft.nl

Nezih Yigitbasi
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
m.n.yigitbasi@tudelft.nl

Alexandru Iosup
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
a.iosup@tudelft.nl

Dick Epema
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands
d.h.j.epema@tudelft.nl

## ABSTRACT

Large-scale distributed computing systems such as grids are serving a growing number of scientists. These environments bring about not only the advantages of an economy of scale, but also the challenges of resource and workload heterogeneity. A consequence of these two forms of heterogeneity is that job runtimes and queue wait times are highly variable, which generally reduces system performance and makes grids difficult to use by the common scientist. Predicting job runtimes and queue wait times has been widely studied for parallel environments. However, there is no detailed investigation on how the proposed prediction methods perform in grids, whose resource structure and workload characteristics are very different from those in parallel systems. In this paper, we assess the performance and benefit of predicting job runtimes and queue wait times in grids based on traces gathered from various research and production grid environments. First, we evaluate the performance of simple yet widely used time series prediction methods and the effect of applying them to different types of job classes (e.g., all jobs submitted by single users or to single sites). Then, we investigate the performance of two kinds of queue wait time prediction methods for grids. Last, we investigate whether prediction-based grid-level scheduling policies can have better performance than policies that do not use predictions.

## Categories and Subject Descriptors

D.4.1 [**Process Management**]: Scheduling;
D.4.7 [**Organization and Design**]: Distributed systems;
D.4.8 [**Performance**]: Modeling and prediction; Simulation

## General Terms

Experimentation, Performance

## Keywords

Predictions, time series prediction methods, grid scheduling, trace-based simulation, performance evaluation

## 1. INTRODUCTION

Over the last decade, an increasing number of scientists have run their workloads on large-scale distributed computing systems such as grids. However, while these systems can be cost-effective and easily scalable, their multi-site and heterogeneous resource structure, and their dynamic and heterogeneous workloads limit the efficient use of the system resources. Moreover, the high variability of the job runtimes and queue wait times make such systems difficult and often frustrating to use for the common user. Prediction methods, and in particular prediction-based scheduling, have been employed to address these problems in parallel production environments, but their use for large-scale distributed systems remains largely unexplored. In this paper we present a systematic investigation of prediction methods with application to grid scheduling.

Grids differ from traditional parallel production environments (PPEs) in both structure and typical use. While a PPE consists of a single (large) tightly coupled supercomputer or a well-interconnected cluster, a grid consists of multiple sites, each of which in the type of grids we consider typically comprising a commodity cluster with a more common network. The arrival process is more bursty in grids, whose workloads often contain many bags-of-tasks [14, 15, 16]. Many grids, and in particular most of those used for production work, only process single-processor jobs; in contrast, in PPE workloads, single-processor tasks are fewer and have much shorter runtimes than parallel jobs [24]. As a combined result of these distinctive grid features, over the long-term, grid workloads usually include many more jobs than those of PPEs, and over the short-term, grids have often long queues comprising single-processor tasks from one or more bags-of-tasks. Both of these phenomena lead to a higher variability of job runtimes and queue wait times than in PPEs, and it is this variability that forms the focus of our investigation.

An extensive body of research has focused on devising and applying prediction methods for such quantities as job runtimes and job queue wait times [1], CPU load [38], re-

source availability [26], and resource failure rates [21] in (large-scale) computer systems such as parallel systems and grids. The aim of such methods is to aid in the efficient scheduling in such systems and to assist users in selecting resources for their jobs. For instance, runtime predictions have been used to improve the performance of backfilling in batch queueing systems [35], and runtime and queue wait time predictions together can guide the decisions of a grid scheduler as to which grid sites to send jobs for execution. What is missing so far from this research is a detailed investigation of the performance of prediction methods for job runtime and queue wait time in grids, and of the benefit of using predictions in grid scheduling. In this paper we fill this gap by applying simple and widely used prediction methods to the job runtimes and queue wait times of nine workload traces of research and production grids in the Grid Workload Archive [17], and by assessing the benefit of using predictions in grid level scheduling via trace-based simulations.

Our investigation is based on three guidelines. First, we target grid systems consisting of multiple clusters in which the processors are managed by space-sharing policies. Secondly, we restrict ourselves to time series prediction methods, which make their predictions based on historical data, usually in the form of the time-ordered set of past observations of, e.g., the job runtimes. Thirdly, we classify jobs in different ways, and apply these methods to the different job classes separately, in the hope to improve the performance of the prediction methods. Among the job classifications we will employ are grouping jobs per grid site, per user, and per user and per site. In this way, we aim to give realistic answers to the following research questions:

- **How accurate are the simple but widely used time series methods in predicting job runtimes in grids, and what is the impact of job classification on the accuracy of these predictions?** We answer these two questions in Section 3, where we assess the accuracy of five time series methods under four job classifications.

- **What is the performance of queue wait time predictors in grids?** We answer this question in Section 4 by evaluating the performance of a *point-valued* predictor that simulates the local scheduling policy with the predicted job runtimes to predict queue wait times of jobs, and by evaluating the performance of methods that predict *upper bounds* for the queue wait times of jobs.

- **Can prediction-based grid scheduling policies perform better than grid scheduling policies that do not use predictions?** We answer this question in Section 5, where we compare three grid-level scheduling policies in a simulated environment. The prediction-based scheduling policy bases its decisions on job runtime and queue wait time predictions (which are either assumed to be perfect or potentially inaccurate), whereas the non-prediction-based policies balance the load on the clusters or prefer faster resources.

## 2. GRID WORKLOAD TRACES

In this study, we use nine grid traces from the Grid Workloads Archive [17]. Each trace consists of ordered job entries according to their submission time which is in UNIX timestamp format. All traces have complete information about the jobs' submission times, runtimes, and requested numbers of nodes. Some of the traces lack other job attributes such as queue wait time, application name, group name, etc. None of the traces contain information about user runtime estimates.

Table 1 summarizes the characteristics of the grid traces that we have used in our work, and contrasts them to those of four traces taken from the Parallel Workloads Archive [34]. The grid traces are gathered from four research grids and five production grids. There are several differences between typical research and production grids. In research grids, the workloads contain both parallel jobs and sequential jobs, and the system utilizations are low (10%-30%), whereas in production grids, the workloads consist solely of sequential jobs, and the system utilizations are much higher (over 60%) [14, 15]. The main differences between grids and PPEs as we observe are the following: grid resources are spread across multiple sites, grid workloads include few or even no parallel jobs, and over the long term, the grid workloads include many more jobs than PPEs (by a factor of 2-20).

An important assumption of this work is that grids exhibit often bursty job arrivals. We show that this is indeed the case in Figure 1, which depicts the numbers of job submissions during five-minute intervals. Eight of the nine grid systems considered in this work have bursty periods; DAS2, the ninth grid system, is similar to DAS3 in nature and thus was omitted from Figure 1. We conclude that in addition to the grid workload characteristics mentioned above, grid workloads are bursty, which provides even more motivation for this study.

## 3. JOB RUNTIME PREDICTIONS

In this section we investigate the performance of job runtime predictions in grids. We first describe in Section 3.1 our methodology. Then, we describe in Section 3.2 our experimental setup, and last we present and discuss in Section 3.3 the experimental results.

### 3.1 Methodology

The methodology we use for runtime predictions consists of three elements: the classifications of jobs we use, the way we simulate grid traces to compute runtime predictions, and the actual runtime prediction methods, which we now describe in turn.

First, the job classification methods create classes according to job attributes such as the execution site of a job, the user submitting it, etc. We consider the following four classification methods for the jobs in a trace:

1. **Site**: The jobs are classified according to the site where they are executed.

2. **User**: The jobs are classified according to the user who submits them, irrespective of the execution site.

3. **User on Site**: The jobs are classified according to both the user and the execution site.

4. **(User + Application Name + Job Size) on Site**: The jobs are classified according to the user, the application name, the job size (i.e., the number of processors employed by the job), and the execution site.

Secondly, in the simulation of a trace, we go sequentially through the trace and we compute for every next job its predicted runtime with the prediction method in place, based on the history consisting of the runtimes of the jobs of the

Table 1: The characteristics of nine grid traces taken from the Grid Workloads Archive and of four traces taken from the Parallel Workloads Archive. The sign "-" denotes missing information.

| | System | | | Trace | | | |
|---|---|---|---|---|---|---|---|
| | Type | Num. of Clusters | Size [CPUs] | Duration [Months] | Size [Tasks] | % of Parallel Jobs | Num. of Users |
| *Grid Workloads Archive Traces [17]* | | | | | | | |
| DAS2 | Research | 5 | 400 | 18 | 1.1M | 66% | 333 |
| GRID5000 | Research | 15 | ∼2500 | 27 | 1.0M | 45% | 470 |
| DAS3 | Research | 5 | 544 | 18 | 2M | 15% | 331 |
| SHARCNET | Research | 10 | 6828 | 12 | 1.2M | 10% | 412 |
| AUVER | Production | 5 | 475 | 12 | 0.4M | 0% | 405 |
| NORDU | Production | 75 | 2000 | 24 | 0.8M | 0% | 387 |
| LCG | Production | - | 24515 | 4 | 0.2M | 0% | 216 |
| NGS | Production | 7 | - | 6 | 0.6M | 0% | 378 |
| GRID3 | Production | 35 | ∼3500 | 18 | 1.3M | 0% | 15 |
| *Parallel Workloads Archive Traces [34]* | | | | | | | |
| CTC SP2, PWA-6 | Production | 1 | 430 | 11 | 0.1M | 56% | 679 |
| SDSC SP2, PWA-9 | Production | 1 | 128 | 24 | 0.1M | 63% | 437 |
| LANL O2K, PWA-10 | Production | 1 | 2048 | 5 | 0.1M | - | 337 |
| SDSC DS, PWA-19 | Production | 1 | 1664 | 13 | 0.1M | 100% | 460 |



(a) Grid5000    (b) DAS3    (c) SHARCNET    (d) AUVER
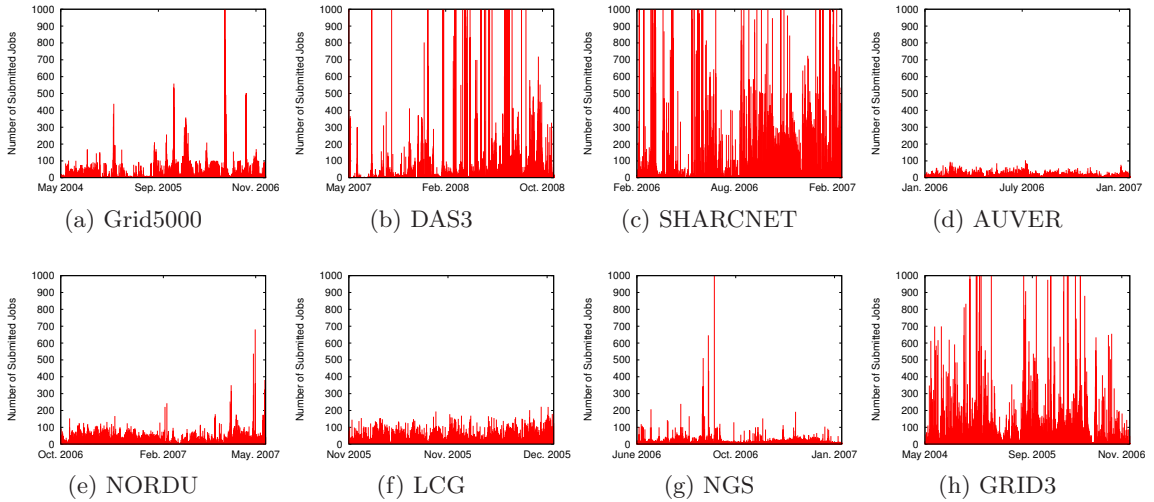
(e) NORDU    (f) LCG    (g) NGS    (h) GRID3

Figure 1: The number of job submissions during five-minute intervals in eight selected grid systems. All systems have bursty periods. The vertical axis is truncated at 1000 for better visibility.

same class that have finished before the job under consideration is submitted. This means that during the simulation, the time series for each of the job classes is created.

Thirdly, for each job, the prediction method predicts the runtime using the time series data of the class the job belongs to that has already been created. We consider the following prediction methods, which are applied to the time series of the runtimes of the jobs on a per-class basis:

1. **Exponential Smoothing (ES)** predicts the runtime as a weighted moving average of the observed job runtimes. A parameter $\alpha$, with $0 \leq \alpha \leq 1$, is used to control the sensitivity of the smoothing. We take $\alpha$ to be equal to 0.5 (e.g., see [6]). We refer to [4] for details.

2. **Running Mean (RM)** predicts the runtime as the mean of all observed job runtimes.

3. **Sliding Median (SM)** predicts the runtime as the median of a sliding window of observed job runtimes. We take 5 as the window size (e.g., see [36]).

4. **Last** predicts the runtime as the last previously observed job runtime.

5. **Last2** [35] predicts the runtime as the average of the last two previously observed job runtimes.

For the first three job classifications introduced above, we evaluate the performance of all prediction methods on all traces. Then, we pick the best method for each trace and run it with the last classification for those traces that include the Application Name attribute (all of the traces except LCG and NORDU). The Job Size attribute is considered only for the research grid traces since only they include parallel jobs.

## 3.2  Experimental Setup

We have used the Grid Workloads Archive tools to process the grid traces. We have extended the tools so that the jobs in the traces are classified according to the classifications described in the previous section.

To evaluate the accuracy of the runtime predictions, we consider the following metrics:
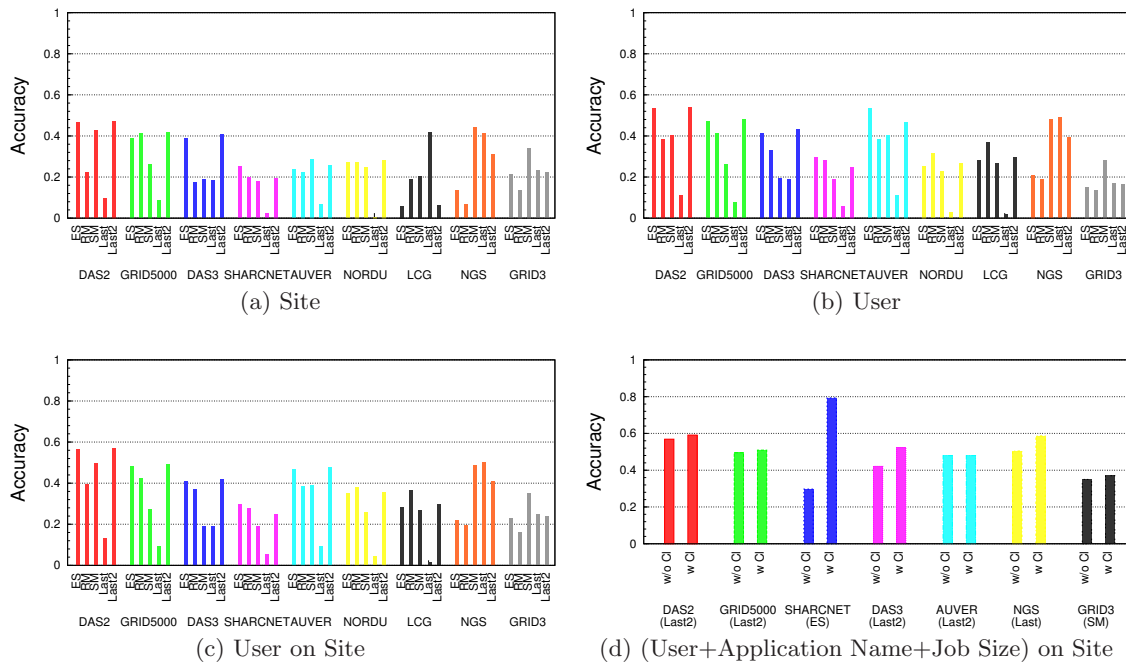
Figure 2: The average accuracy of the job runtime prediction methods on the grid traces under four job classifications.

1. The **accuracy**, which is defined as in [35]:

$$accuracy = \begin{cases} 1 & \text{if } P = T_r, \\ T_r/P & \text{if } P > T_r, \\ P/T_r & \text{if } P < T_r, \end{cases} \quad (1)$$

   where $P$ is the predicted job runtime and $T_r$ is the actual job runtime.

2. The **absolute prediction error** is the absolute difference between the predicted and the actual runtime.

## 3.3 Results

For each of the nine traces, Figures 2.(a), (b), and (c) present the average accuracy of the prediction methods under the Site, the User, and the User on Site classifications, respectively. Figure 2.(d) shows the average accuracy of the best methods under the (User + Application Name + Job Size) on Site classification (**w/o Cl**: Best result from the other classifications, **w Cl**: Result with this classification.).

As the historical data gets more specific, that is, going from Site to (User + Application Name + Job Size) on Site, the accuracy of the job runtime predictions increases significantly. In particular, SHARCNET yields an outstanding job runtime accuracy with the most specific classification. For the traces of DAS2, DAS3, and GRID5000, Last2 performs better than the other prediction methods for all classifications. For the other traces, we do not observe such a dominant method, and even the best method for a trace differs among the classifications. The results suggest that grid systems or even grid sites should have their own specific prediction methods, since they may have different user behaviors and different job and system characteristics.

Figure 3 presents the cumulative distributions of the accuracy and the absolute error only for the best results (the method and classification that give the best accuracy for a trace) for the traces of the research and production grids

separately. We observe that in most of the cases, the job runtimes are predicted more accurately and with lower absolute errors in research grids than in production grids. The possible reasons include longer job runtimes and higher utilizations in production grids. For SHARCNET, Almost 70% of the predictions have high accuracy (i.e., above 0.9), while for the other traces this percentage ranges between 20 to 30. Among the production grid traces, we see that NGS and GRID3 exhibit a higher prediction accuracy and a lower absolute prediction error.

All in all, we find the job runtime prediction accuracy to be low and the absolute prediction error to be high, even for the best results (except for SHARCNET). There are several reasons for this poor performance. The first is the occurrence of burst submissions that we observe in grids (e.g., see Figure 1); the same prediction error is made for all the jobs submitted together or relatively close in time. Even though these jobs could be similar in terms of runtime, they do not affect the predictions before they finish execution. A second reason is the (lack of) stationarity of a time series. For good predictability, a time series should be stationary [4], that is, it should have a constant long-term mean and variance. We have performed several experiments using the Augmented-Dickey-Fuller[1] (ADF) test [11] for checking stationarity on some of the time series that we use in this work; unsurprisingly, we have found the time series to be non-stationary.

## 4. QUEUE WAIT TIME PREDICTIONS

In this section we investigate the performance of queue wait time predictions in grids. In Section 4.1 we evaluate the performance of a point-valued predictor that simulates the local scheduling policy with predicted job runtimes to predict job queue wait times. In Section 4.2 we evaluate

---

[1]We have obtained the tool for the ADF-test from http://www.web-reg.de/adf_addin.html
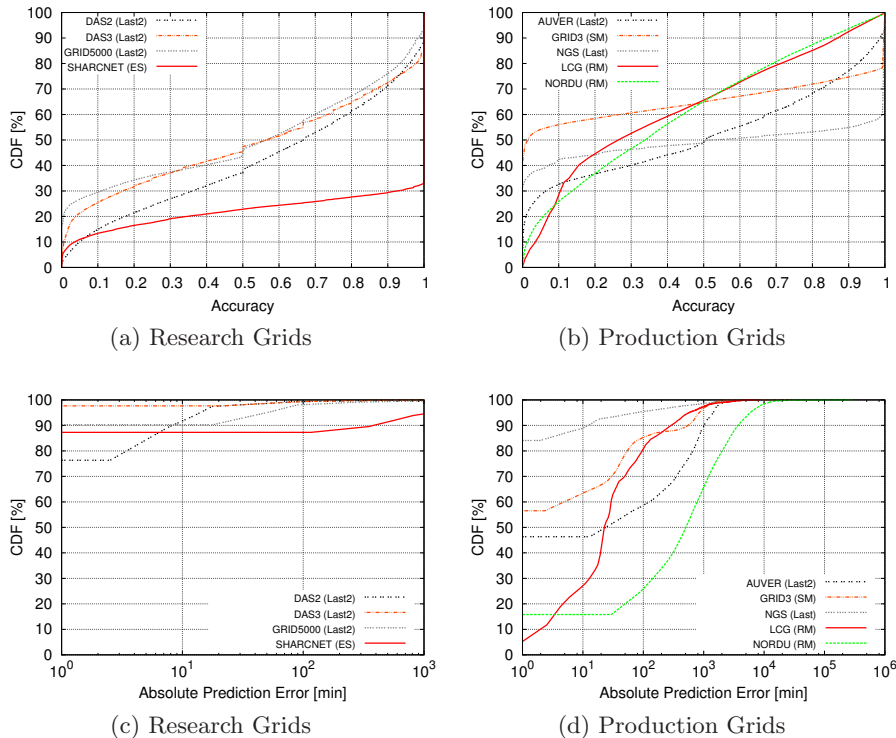
Figure 3: The cumulative distribution functions of the accuracy and absolute prediction error for the best (method+classification) results for job runtimes. The horizontal axis for the bottom row has a logarithmic scale.

Table 2: The performance of the point-valued predictor of queue wait time for jobs that have non-zero wait times (indicated by "non-zero jobs")

| | | Without Correction | | With Correction | |
|---|---|---|---|---|---|
| Grid-Site | % non-zero jobs | Avg. Accuracy | AWPE [min] | Avg. Accuracy | AWPE [min] |
| DAS2-FS1 | 20 | 0.60 | 110 | 0.64 | 121 |
| DAS2-FS3 | 35 | 0.54 | 307 | 0.55 | 292 |
| DAS3-FS3 | 80 | 0.30 | 364 | 0.35 | 350 |
| DAS3-FS4 | 70 | 0.51 | 451 | 0.60 | 442 |
| Grid5K-G1/S1/C3 | 60 | 0.51 | 322 | 0.63 | 256 |
| Grid5K-G1/S6/C1 | 10 | 0.56 | 852 | 0.61 | 653 |
| AUVER-clrlcgce01 | 20 | 0.57 | 203 | 0.64 | 180 |
| AUVER-clrlcgce03 | 67 | 0.63 | 190 | 0.69 | 172 |

the performance of two non-parametric statistical methods that predict upper bounds for queue wait times with a specified confidence level. Such non-parametric methods have the advantage of obviating the need to know the internal operation of local scheduling policies in predicting the queue wait times. We use the traces of the DAS2, DAS3, GRID5000, and AUVER grids. These are the systems/traces of which we know their characteristics to model in our simulations in Section 4.1, and that contain the queue wait time data that we need in the simulations in Section 4.2.

## 4.1 Point-Valued Predictions

In our simulation model, each site has a Local Resource Manager (LRM) that employs the FCFS policy without back-filling. We assign jobs to their original execution sites. A point-valued predictor runs on each site, and computes a queue wait time prediction for each submitted job in a two-step process. First, the runtimes of all queued and running

jobs are predicted with the Last2 method and the (User + Application Name + Job Size) on Site classification that is described in Section 3.1. Then, the scheduling policy of the LRM is simulated with the predicted job runtimes to determine when the new job will start. Whenever the runtime of a job turns out to be under-predicted, its predicted runtime is doubled until the predicted value is larger than the actual runtime.

We also consider a prediction correction mechanism in which upon completion of a job, the predicted runtimes of both the queued and the running jobs that belong to the same class as the completed job are updated with the Last2 method to include the runtime of the completed job in the computation. We perform the experiments with and without this correction mechanism.

Figure 4 shows the cumulative distribution functions of the accuracy and the absolute prediction error of the point-valued queue wait time predictor for DAS3 and AUVER; for
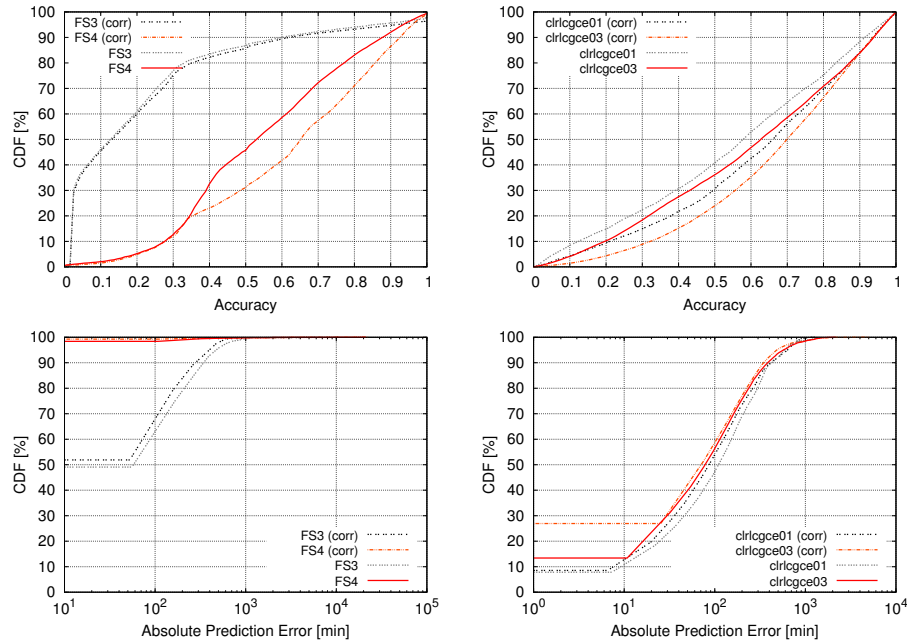
Figure 4: The cumulative distribution functions of the accuracy and absolute prediction error of the queue wait time predictions for the DAS3 (left) and the AUVER (right) traces with the Last2 method for runtime predictions; (corr) denotes that correction mechanism is applied to the job runtime predictions. The horizontal axis has a logarithmic scale for the bottom row.

Table 3: The accuracy of queue wait time prediction methods giving upper bounds.

| | BMBP | | | | |
| --- | --- | --- | --- | --- | --- |
| Grid-Site | AWPE [min] | Avg. Accuracy | under-predictions | perfect-predictions | over-predictions |
| DAS2-FS1 | 16 | 0.50 | 8% | 9% | 83% |
| DAS3-FS4 | 26 | 0.41 | 15% | 4% | 81% |
| AUVER-clrlcgce01 | 376 | 0.20 | 12% | 1% | 87% |
| GRID5K-G1/site1/c3 | 31 | 0.72 | 20% | 0% | 80% |

| | Chebyshev | | | | |
| --- | --- | --- | --- | --- | --- |
| Grid-Sites | AWPE [min] | Avg. Accuracy | under-predictions | perfect-predictions | over-predictions |
| DAS2-FS1 | 52 | 0.21 | 8% | 0% | 92% |
| DAS3-FS4 | 101 | 0.23 | 7% | 1% | 82% |
| AUVER-clrlcgce01 | 1236 | 0.10 | 7% | 0% | 93% |
| GRID5K-G1/S1/C3 | 1093 | 0.24 | 16% | 0% | 84% |

clarity, we only present the results for the two sites of each grid system to which most of the jobs have been submitted. Table 2 presents the average values of the metrics for all grids (AWPE refers to average absolute queue wait time prediction error). In the results, we only consider the jobs that have non-zero wait times.

We find the overall accuracy of the point-valued predictor to be low, and the average absolute queue wait time prediction error to be high due to possibly inaccurate job runtime predictions. The prediction error in the queue wait times is accumulated because the predictor simulates the local scheduling policy with inaccurately predicted runtimes. There is an improvement ranging from 1 to 10% in average accuracy and from 1 to 16% in average absolute prediction error when the prediction correction mechanism is applied.

## 4.2 Upper-Bound Predictions

In this section we assess the accuracy of two upper-bound methods for queue wait time predictions by means of trace-based simulations. These methods are the Binomial Method Batch Predictor (BMBP) [3] and a predictor that makes use of Chebyshev's inequality [33]. In our analysis, we use the wait times of the jobs that are completed in order to predict the wait time of a new job. We do not simulate local scheduling policies since we use real wait time and runtime data (from the traces). To evaluate the prediction methods, we use the average accuracy, the average absolute prediction error, and the number of under-predictions, perfect-predictions, and over-predictions as metrics.

BMBP predicts an upper-bound with a specified quantile and confidence level. It uses the history of job queue wait times, and estimates the quantile of the wait time distribution with the specified confidence level. It employs a change-point detection method in order to take only a stationary part of the time series into account. By detecting the change-points in the time series, BMBP trims the historical wait time data. BMBP also clusters the jobs based
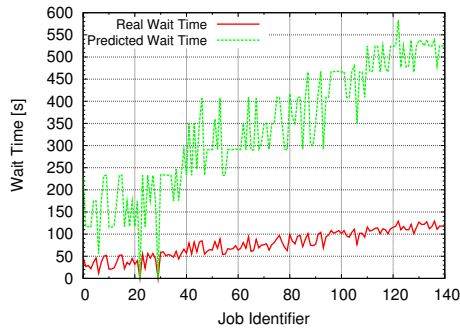
Figure 5: Real and predicted queue wait times for a burst submission case taken from DAS3-FS4.

on the numbers of processors they request, hence it uses the historical data of similar jobs when making predictions.

Chebyshev's inequality states that regardless of the underlying distribution, the probability of a random variable differing from its mean by more than $k$ standard deviations is less than or equal to $1/k^2$. We have implemented a predictor that uses this inequality; it calculates the mean and the standard deviation of the wait time data of the completed jobs to predict an upper bound for the wait time of a new job with a specified confidence level (e.g., $\mu + 2\sqrt{5}\sigma$ is the predicted wait time with a 95% confidence level, where $\mu$ is the mean and $\sigma$ is the standard deviation of the historical wait time data). We trim and update the historical data in a similar way as explained in [3].

In our analysis of the BMBP method, we use the BMBP trace-based simulator [2], and for Chebyshev's inequality method we use our own tools. For BMBP, we consider a quantile and confidence level of 95%, and we use 10% of the data for training. Similarly, for the predictor that uses Chebyshev's inequality, we consider a confidence level of 95%.

The results are presented for a single site of each trace in Table 3. The number of over-predictions when using Chebyshev's inequality is larger than when using BMBP, whereas the accuracy of BMBP is higher. There is a trade-off between the accuracy and the tightness of the upper-bound. Both of these methods fail when the jobs arrive in bursts, as the methods use the same predicted wait time value for all jobs in a burst.

We claim that user runtime estimates, if available, can also be used in predicting upper bounds for queue wait times. To show this, we use a simple model for user runtime estimates that is proposed by Mu'alem and Feitelson [13]. The model assumes that a job's estimate is uniformly distributed within $[R, 5R]$, where $R$ is the job's actual runtime. We use the runtime estimates of users together with the FCFS policy (to guarantee upper bounds) to predict wait times. Figure 5 shows the real wait time values and the ones predicted with this approach for a bursty period of the DAS3-FS4 site.

# 5. THE PERFORMANCE OF PREDICTION-BASED GRID SCHEDULING

In this section, we assess whether it is beneficial to use predictions for scheduling in grids. To this end, we perform a set of simulations using workloads from the DAS3 and AUVER grids to investigate whether prediction-based

---

[2]We have obtained the simulator from the Network Weather Service website http://nws.cs.ucsb.edu.

Table 4: The size of the clusters in DAS3 and AUVER grids.

| DAS3 | | AUVER | |
|---|---|---|---|
| Cluster (Original Name) | Size | Cluster (Original Name) | Size |
| C1 (FS0) | 85 | C1 (clrlcgce01) | 112 |
| C2 (FS1) | 32 | C2 (clrlcgce02) | 84 |
| C3 (FS2) | 41 | C3 (clrlcgce03) | 186 |
| C4 (FS3) | 68 | C4 (iut15) | 38 |
| C5 (FS4) | 46 | C5 (opgc) | 55 |

Table 5: The workload characteristics used for assessing the performance of prediction-based grid scheduling.

| Trace | Period | Number of Jobs | Avg. Utilization |
|---|---|---|---|
| DAS3 | July-Oct. 2008 | $\sim$220,000 | $\sim$30% |
| AUVER | Aug.-Nov. 2006 | $\sim$90,000 | $\sim$70% |

grid-level scheduling improves performance over traditional grid-level scheduling policies. In Section 5.1 we explain the experimental setup, in Sections 5.2 and 5.3 we describe the scheduling policies and the performance metrics we use in our simulations, respectively, and in Section 5.4 we present and discuss the experimental results.

## 5.1 The Experimental Setup

For our experiments, we have modeled two multi-cluster grid environments, the DAS3 (research) and the AUVER (production) grids, using our event-based grid simulator DGSim [19]. Table 4 shows the sizes of the clusters of each system. In DAS3, the processing speeds of the compute nodes differ among clusters, but in AUVER, the nodes are homogeneous in terms of processing speeds. To model the processor heterogeneity across the clusters of the DAS3, we employ the SPEC CPU benchmark model for job runtimes [32], that is, the time it takes to finish a job is inversely proportional to the performance of the node it runs on.

In our model, each cluster has its own LRM, and so its own local queue to which jobs arrive, and a global central scheduler with a global queue operates on top of the cluster LRMs. The jobs are submitted to the global scheduler, which decides in which cluster a job is going to run based on one of the scheduling policies that are explained in Section 5.2. Irrespective of the policy in operation, the global scheduler considers all jobs in its queue as the eligible set for scheduling. The LRMs of the clusters employ the FCFS policy without backfilling. Once started, tasks run to completion, so we do not consider task preemption or task migration during execution. For the experiments with prediction-based policies, a prediction service runs on each of the clusters in order to respond to the queries issued by the central scheduler regarding the predicted completion time of a job; i.e., the sum of the predicted queue wait time of a job and its predicted runtime.

In our simulations, we consider the busiest three-month period from the trace of each system and submit it as the workload. Table 5 shows the properties of the workloads that we have used in our experiments.

## 5.2 Scheduling Policies

We compare the performance of the following policies which we find representative for many other prediction-based and traditional policies proposed in the grid scheduling literature:
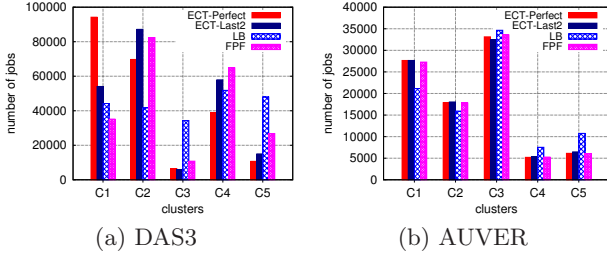
(a) DAS3          (b) AUVER

Figure 7: The distribution of jobs across the clusters of DAS3 and AUVER.

- The **Earliest Completion Time** (ECT) [25] is a Gantt chart-based scheduling policy [5] that submits each job to the cluster that leads to the earliest completion time possible, taking into account the clusters' queues. We consider two kinds of prediction information leading to two variations of this policy. **ECT-Perfect** policy is a theoretical omniscient policy whose predictions are always given with perfect accuracy (equal to 1). In contrast, **ECT-Last2** uses the point-valued predictor defined in Section 4.1 (with corrections); hence, the predictions of **ECT-Last2** may be inaccurate.
- **Load Balancer** (LB) submits each job to the least-loaded cluster, where load is defined as the total processor requirement of all jobs running or queued in the cluster normalized by the cluster size.
- **Fastest Processor First** (FPF) [18] submits each job to the cluster that has the fastest compute nodes among the clusters that have enough idle nodes to accommodate the job. Different from the policies mentioned above, FPF does not forward jobs as long as there are not enough idle nodes in any cluster; therefore, jobs are only queued in the global queue.

## 5.3 Performance Metrics

To assess the performance of the scheduling policies, we use the following traditional metrics:

- The **Queue Wait Time** of a job is the time elapsed from the submission of the job until the start of its execution.
- The **Response Time** of a job is the sum of its queue wait time and its runtime.
- The **Bounded Slowdown** [12] of a job is defined as

$$\max\left(1, \frac{T_w + T_r}{\max(\tau, T_r)}\right), \qquad (2)$$

where $T_w$ and $T_r$ denote the queue wait time and the runtime of the job, respectively, and $\tau$ denotes the threshold for the job runtimes. The bounded slowdown eliminates the emphasis on short jobs due to having the runtime in the denominator. In our analysis we have used a threshold value of 60 seconds.

## 5.4 Results

The cumulative distribution functions of the queue wait time, the response time, and the bounded slowdown are shown in Figure 6, and their averages are shown in Table 6.

We find that in DAS3, the prediction-based scheduling policies (ECT-Perfect and ECT-Last2) perform better than

Table 6: The performance of the three scheduling policies.

| DAS3 | ECT-Perfect | ECT-Last2 | LB | FPF |
|---|---|---|---|---|
| Avg. Res. Time [s] | 1320 | 1400 | 4318 | 1911 |
| Avg. Wait Time [s] | 105 | 186 | 3061 | 681 |
| Avg. Boun. Slowd. | 1.7 | 1.6 | 80 | 26 |

| AUVER | ECT-Perfect | ECT-Last2 | LB | FPF |
|---|---|---|---|---|
| Avg. Res. Time [s] | 40951 | 41003 | 40959 | 41334 |
| Avg. Wait Time [s] | 6515 | 6574 | 6534 | 6898 |
| Avg. Boun. Slowd. | 48 | 43.6 | 48 | 50.88 |

their traditional counterparts (LB and FPF), with especially the LB policy having very poor performance. It turns out that with LB, approximately 5% of the jobs have a queue wait time of more than 10,000 s. On the other hand, with the FPF policy, a small number of jobs suffer from queue wait times as high as 10,000 s. In contrast, for AUVER, all policies have similar performance. LB seems to perform slightly better than FPF in AUVER, which suggests that it can be a candidate for highly utilized systems when prediction-based policies are not considered. In general, the performance of the considered policies is worse for the AUVER grid, which is probably due to the higher utilization of the considered period compared to the DAS3 (see Table 5).

Since the ECT-Perfect and ECT-Last2 policies have similar performance, both in the DAS3 and the AUVER experiments, we conclude that more accurate predictions do not necessarily imply a better performance of grid scheduling. A similar result was previously obtained when using predictions for improving backfilling performance in parallel production systems [35]. The similar performance results of ECT-Perfect and ECT-Last2 may be due to similar scheduling decisions; in Figure 7 we see that the clusters receive similar numbers of jobs when using these policies. Therefore, we claim that the main concern of a prediction-based grid scheduling policy is the correct prediction of the cluster that will finish a job first, rather than perfect prediction accuracy of the job's completion time.

## 6. RELATED WORK

Previous work on predictions has mainly focused on proposing novel prediction methods [9, 31, 36, 3, 8], on enhancing existing methods [7, 29], and on making use of predictions in space-shared parallel environments [35, 23]. In contrast, in this work we focus on the performance of job runtime and queue wait time predictions in grid environments.

In addition to various complex solutions proposed for job runtime predictions such as analytical benchmarking/code profiling [28, 37, 20], genetic algorithms [30, 31], and instance-based learning [22], simple time series methods have also received great attention from the community due to their advantages such as ease of implementation and speed of delivering prediction results. In [7, 29], exponential smoothing is used for predicting the runtimes of jobs. Dobber et al. [8] present a survey on prediction methods for job runtimes on space-shared processors, and they propose a new prediction method, called Dynamic Exponential Smoothing, which uses exponential smoothing and adapts dynamically to peaks and level changes in the job runtimes. Feitelson et al. [35] show that even a simple time series prediction method improves backfilling performance significantly when system-generated predictions replace user-estimated runtimes. The Network
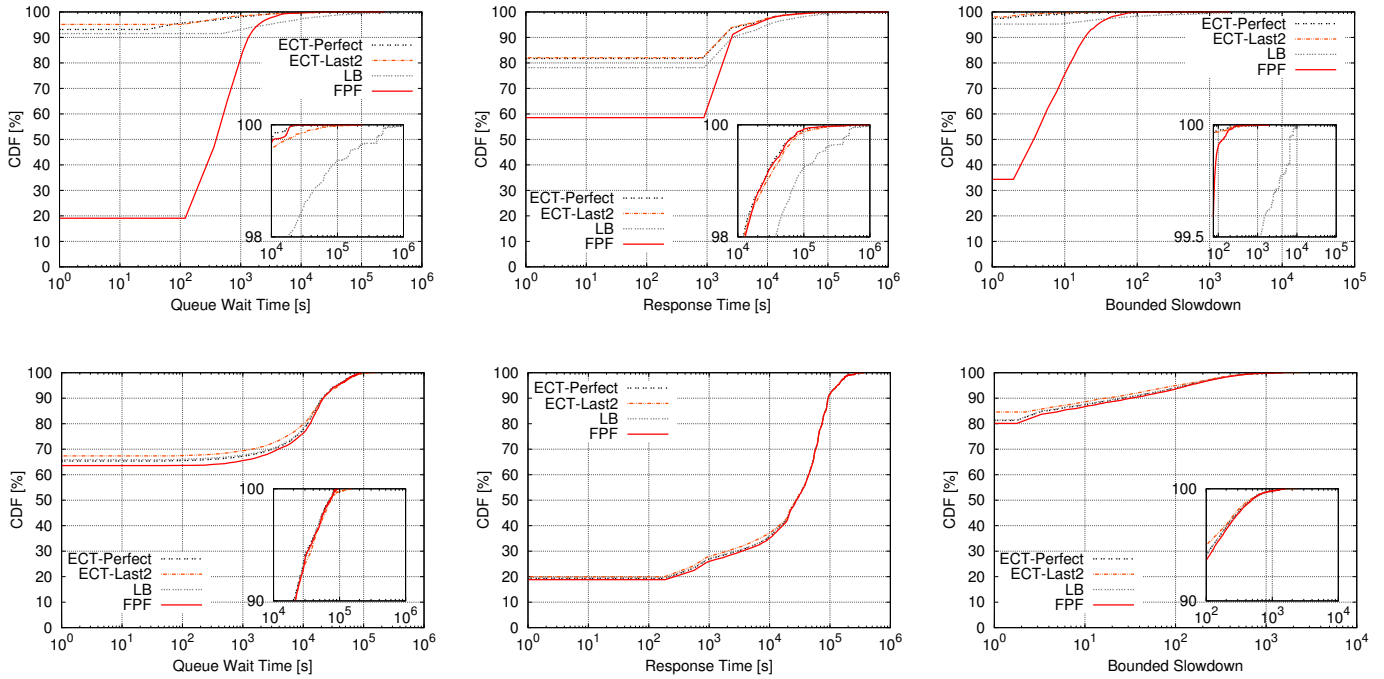
Figure 6: The cumulative distribution functions of the queue wait time, the response time, and the bounded slowdown for DAS3 (top) and AUVER (bottom). The horizontal axis has a logarithmic scale.

Weather Service (NWS) [36] is a well-known prediction service which is used to predict the performance of computational grid resources, with simple time series prediction methods. NWS tracks the accuracy of all its predictors, and dynamically changes the prediction method to the one that gives the highest accuracy.

The problem of predicting queueing delays of jobs in high performance computing settings has also received constant attention from the research community. In [2], model-fitting is used to model machine availability in desktop and enterprise grid computing environments. To estimate when a cluster of a given size will be available and hence the run-time of the job at the head of the queue, Downey [10, 9] explore a log-uniform distribution to model the remaining lifetimes of the jobs. This work shows that the queue wait time of jobs can be predicted if the runtimes of jobs and the scheduling algorithm are known. Similarly, in [31] the queue wait time of a job is predicted under the conditions that the job runtimes and the local scheduling algorithm are known. In this work, the authors use a template-based approach to cluster the jobs, and then perform searches based on genetic algorithms. Wolski et al. [27] propose QBETS, which is a non-parametric time series method to predict bounds on the queue wait times of individual jobs in space-shared parallel environments.

## 7. CONCLUSIONS

Grids have highly variable job runtimes and job queue wait times, which can lead to degraded system performance and to decreased usability for the common scientist. Prediction methods are commonly used to alleviate these problems in traditional parallel production environments, but the large body of existing research cannot be directly applied to grids. In this paper we have studied job runtime and queue wait time prediction methods and their application in grid

scheduling. First, we have studied through trace-based analysis the accuracy of well-known, simple time series prediction methods when predicting job runtimes, and the impact of job classification on the accuracy of job runtime predictions. We found that the low accuracy of time series prediction methods for grids can be improved significantly by the use of such classifications. Secondly, we have analyzed the performance of queue wait time predictors. We have shown that current solutions for queue wait time predictions that give upper-bounds cannot handle the common grid workload characteristic of burst submissions. Thirdly, we have compared with trace-based simulations several prediction-based and traditional grid scheduling policies in order to investigate the impact of predictions on the performance of grid-level scheduling. We have found that a better accuracy of the predictions does not imply a better performance of grid scheduling, and that for the highly utilized production grids the investigated policies perform similarly to each other.

For the future, we plan to investigate how existing prediction methods can be improved and adapted to computational grids, where burst submissions are common, and the historical runtime and wait time data are non-stationary.

## Acknowledgments

## 8. REFERENCES

[1] F. Berman, R. Wolski, H. Casanova, and W. Cirne. Adaptive computing on the grid using AppLeS. *IEEE TPDS*, 14(4):369–382, 2003.

[2] J. Brevik, D. Nurmi, and R. Wolski. Automatic methods for predicting machine availability in desktop Grid and peer-to-peer systems. In *CCGRID*, pages 190–199, 2004.

[3] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *PPoPP*, pages 110–118, 2006.

[4] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, March 2002.

[5] H. Casanova, F. Berman, G. Obertelli, and R. Wolski. The AppLeS parameter sweep template: User-level middleware for the grid. In *SC Conference*, pages 60–79, 2000.

[6] M. Dobber, G. Koole, and R. V. D. Mei. Dynamic load balancing for a grid application. In *HiPC*, pages 342–352, 2004.

[7] M. Dobber, G. Koole, and R. van der Mei. Dynamic load balancing experiments in a grid. In *CCGRID*, pages 1063–1070, 2005.

[8] M. Dobber, R. van der Mei, and G. Koole. A prediction method for job runtimes on shared processors: Survey, statistical analysis and new avenues. *Perform. Eval.*, 64(7-8):755–781, 2007.

[9] A. B. Downey. Predicting Queue Times on Space-Sharing Parallel Computers. In *IPPS*, pages 209–218, 1997.

[10] A. B. Downey. Using Queue Time Predictions for Processor Allocation. In *JSSPP*, pages 35–57, 1997.

[11] G. Elliott, T. J. Rothenberg, and J. H. Stock. Efficient tests for an autoregressive unit root. *Econometrica*, 64(4):813–836, 1996.

[12] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *JSSPP*, pages 1–34, 1997.

[13] D. Feitelson and A. Weil. Utilization and predictability in scheduling the ibm sp2 with backfilling. In *PPS*, pages 542–561, 1998.

[14] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *GRID*, pages 262–269, 2006.

[15] A. Iosup, D. Epema, C. Franke, A. Papaspyrou, L. Schley, B. Song, and R. Yahyapour. On grid performance evaluation using synthetic workloads. In *JSSPP*, pages 232–255, 2006.

[16] A. Iosup, M. Jan, O. Sonmez, and D. Epema. The Characteristics and Performance of Groups of Jobs in Grids. In *Euro-Par*, pages 382–393, 2007.

[17] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema. The grid workloads archive. *FGCS*, 24(7):672–686, 2008.

[18] A. Iosup, O. Sonmez, S. Anoep, and D. Epema. The performance of bags-of-tasks in large-scale distributed systems. In *HPDC*, pages 97–108, 2008.

[19] A. Iosup, O. Sonmez, and D. Epema. DGSim: Comparing Grid Resource Management Architectures through Trace-Based Simulation. *LNCS*, 5168:13–25, 2008.

[20] M. A. Iverson and G. J. Follen. Run-time statistical estimation of task execution times for heterogeneous distributed computing. In *HPDC*, pages 263–270, 1996.

[21] W. Kang and A. Grimshaw. Failure prediction in computational grids. In *ANSS*, pages 275–282, 2007.

[22] N. H. Kapadia, J. A. B. Fortes, and C. E. Brodley. Predictive application-performance modeling in a computational grid environment. In *HPDC*, pages 47–54, 1999.

[23] B.-D. Lee and J. M. Schopf. Run-Time Prediction of Parallel Applications on Shared Environments. In *Cluster*, volume 0, pages 487–582, 2003.

[24] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[25] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *HCW*, pages 30–44, 1999.

[26] F. Nadeem, R. Prodan, T. Fahringer, and A. Iosup. A Framework For Resource Availability Characterization And Online Prediction in the Grids. In *CoreGRID Integration Workshop*, pages 209–224, 2008.

[27] D. C. Nurmi, J. Brevik, and R. Wolski. QBETS: Queue Bounds Estimation from Time Series. In *SIGMETRICS*, pages 379–380, 2007.

[28] D. Pease, A. Ghafoor, I. Ahmad, D. L. Andrews, K. Foudil-Bey, T. E. Karpinski, M. A. Mikki, and M. Zerrouki. Paws: A performance evaluation tool for parallel computing systems. *IEEE Computer*, 24(1):18–29, 1991.

[29] K. H. Shum. Adaptive Distributed Computing through Competition. In *ICCDS*, page 220, 1996.

[30] W. Smith, I. Foster, and V. Taylor. Predicting Application Run Times Using Historical Information. In *JSSPP*, pages 122–142, 1998.

[31] W. Smith, V. Taylor, and I. Foster. Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance. In *JSSPP*, pages 202–219, 1999.

[32] SPECCPU Team. SPEC CPU2006. Standard Performance. http://www.spec.org/cpu2006/.

[33] H. Stark and J. W. Woods. *Probability, random processes, and estimation theory for engineers*. Prentice-Hall, Inc., 1986.

[34] The Parallel Workloads Archive Team. The parallel workloads archive logs, Jan. 2009. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

[35] D. Tsafrir, Y. Etsion, and D. G. Feitelson. Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE TPDS*, 18(6):789–803, 2007.

[36] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. In *SIGMETRICS*, pages 575–611, 2006.

[37] J. Yang, I. Ahmad, and A. Ghafoor. Estimation of execution times on heterogeneous supercomputer architectures. In *ICPP*, pages 219–226, 1993.

[38] Y. Zhang, W. Sun, and Y. Inoguchi. Predict task running time in grid environments based on cpu load predictions. *FGCS*, 24(6):489–497, 2008.