Predicting Job Start Times on Clusters

Hui Li[*†] David Groep[†] Jeff Templon[†] Lex Wolters[*]
[*] Leiden Institute of Advanced Computer
Science, Leiden University
The Netherlands
{hli,llexx}@liacs.nl
[†] National Institute for Nuclear and High
Energy Physics (NIKHEF)
The Netherlands
{davidg,templon}@nikhef.nl

Abstract

In a Computational Grid which consists of many computer clusters, job start time predictions are useful to guide resource selections and balance the workload distribution. However, the basic Grid middleware available today either has no means of expressing the time that a site will take before starting a job or uses a simple linear scale. In this paper, we introduce a system for predicting job start times on clusters. Our predictions are based on statistical analysis of historical job traces and simulation of site schedulers. We have deployed the system on the EDG (European DataGrid) production cluster at NIKHEF. The experimental results show that acceptable prediction accuracy is achieved to reflect real site states and site-specific scheduling policies. We find that the average error of our job start time predictions is 18.9 percent of the average job queue wait time and this is around 20 times smaller than the average prediction error using the EDG solution.

1. Introduction

Job start time predictions are useful for resource selections in the Grid, provided that acceptable accuracy is achieved to reflect real site states. However, the basic Grid middleware available today either has no means of expressing the time that a site will take before starting a job or uses a simple linear scale. In the European DataGrid [1], for instance, every computing resource (corresponding to a batch queue) publishes one single job start time, which is based on the user specified wall clock times and the number of queued jobs. This seems to be a reasonable approach, except that experience shows that it cannot deliver acceptable accuracy for the *resource broker* [2] to make proper decisions. Several limitations are found in this approach. Firstly, the user specified wall clock times are generally much larger than the actual job run times, which results in large prediction errors. Secondly, it assumes that FCFS (First Come First Serve) scheduling is used at all sites. This is not valid since sites have different scheduling systems and they are generally more sophisticated than FCFS. Thirdly, every site has its own set of scheduling policies and jobs from different Virtual Organizations (VOs) would most likely have different job start times. Therefore publishing single job start time estimates in the Grid Information Service [3] is not sufficient and we need more sophisticated and detailed job start time predictors.

In this paper, we present a job start time prediction system for clusters. Our system is based on statistical predictions of job run times and simulations of schedulers. We obtain job start time predictions via a chain of steps: 1) historical job information is used to predict execution times of jobs currently running and queued at the site; 2) a scheduler simulation is performed, along with the predicted job run times, to determine how long it will take before a newly-submitted job will start execution; 3) predicted job start times are published to the Grid Information Service in accordance to the scheduling policies defined at the site. We have deployed the system on the NIKHEF EDG production cluster. We find that the average error of our job start time predictions is 18.9 percent of the average queue wait time and it significantly improves the originally implemented EDG solution. We also evaluated our job run time prediction technique on clusters subject to a more diverse workload. The average prediction errors range from 13 to 35 percent of the average job run times.

The rest of the paper is organized as follows: Section 2 describes our technique to predict job run times and it is evaluated using workload traces recorded on three selected clusters. Section 3 describes how we simulate the site scheduling system and our technique to improve the simulation performance. Section 4 describes the system design, the idea of incorporating site scheduling policies and experimental results when deploying the system on the NIKHEF EDG cluster. In Section 5 conclusions are presented and future work is discussed.

Cluster	Location	OS	LRMS	CPUs	Period	Job entries
EDG production	NIKHEF	Linux	PBS	20	01/2003 - 04/2003	11537
DAS-2	VU	Linux	PBS	144	01/2003 - 04/2003	40096
DAS-2	UvA	Linux	PBS	64	01/2003 - 04/2003	5857

Table 1. Characteristics of clusters and job traces (LRMS - Local Resource Management System).

2. Predicting job run times

The first step of obtaining job start time predictions is to predict job run times. This part of work is based on statistical techniques [4, 5, 6], in which predictions are generated by applying statistical methods on historical job traces.

2.1. Related work

In [4, 6], jobs in historical traces are categorized according to their attributes (user name, executable name, etc). The *templates*, which are defined as a set of job attributes, generate categories to which jobs can be assigned. Jobs that fall into the same category are considered similar and statistical methods such as *mean* or *linear regression* are applied to generate run time predictions. Various approaches differ in the set of job attributes used and their template definitions. A comparison of these techniques is available in [6].

Compared with previous approaches, we go one step further towards prediction generation. In [6], among all estimates produced by the set of chosen templates and estimators, the one with the smallest confidence interval is selected as the prediction. In our approach, we evaluate different techniques to select estimates. These techniques include choosing an estimate based on previous prediction errors, or combining the estimates to produce new predictions. Finally, the technique with the smallest average prediction error is selected to implement the job run time predictor.

Our experiments are mainly based on the NIKHEF EDG [7] production cluster. For comparative studies, we also use traces on DAS-2 [8] clusters at UvA (Universiteit van Amsterdam) and VU (Vrije Universiteit Amsterdam). Characteristics of these clusters and workload traces are given in Table 1.

2.2. Template definition and evaluation

The first step of our approach is to define a suitable set of templates and evaluate them quantitatively using historical traces. For our traces, we find that *group name* (G), *user name* (U), *queue name* (Q), *executable name* (E) and *number of CPUs allocated* (N) are key job attributes that can be used for job categorization. With these attributes we can theoretically define 32 (2^5) different templates. Genetic algorithms can be applied to search for templates with the smallest prediction errors, as is investigated in [6]. In our case we define the template space by heuristics, which can be obtained from the statistical properties of the historical traces. This results in the following templates, which forms a representative job classification and categorizes jobs from coarse to fine granularity:

[G], [G, U], [G, U, Q],
[G, U, E], [G, U, E, N], [G, U, Q, E, N].

We also selected two candidate statistical estimators for quantitative evaluation. They are:

WM(n) An AR(*n*) (Auto Regressive) model with all coefficients set to 1/*n*. This predicts the next sequence value to be the average of previous *n* values, a simple *Windowed Mean*. AR is one of the Time Series Analysis models, which are investigated extensively in Dinda's work for host load prediction [9].

LR(n) *Linear Regression* [10, 11], where *n* is the number of previous values used for estimation.

We conduct the quantitative evaluation by actually predicting execution times of historical jobs using traces given in Table 1. Results are shown in Figure 1, 2 and 3.

Firstly we evaluate the results on the NIKHEF EDG production cluster. As can be seen in Figure 1, average prediction errors become smaller as the number of previous values used (n) decreases, both for LR and WM estimators. We select two estimators with the smallest prediction errors, which are WM(1) and LR(5). With respect to templates, we eliminate those with the same number or more attributes but produce no better results and keep [G], [G, U] and [G, U, Q] as our templates for generating predictions. This result is consistent with the statistical properties of NIKHEF EDG traces, where attribute E (Executable name) and N (Number of CPUs allocated) provide no extra information for categorization. Template [G] and [G, U] should be kept since they can provide estimations in case that no dedicated historical data is available in finer grain templates (e.g. [G, U, Q]). For the NIKHEF EDG cluster, the predictors are combinations of the selected templates and the selected estimators: {[G, U, Q], LR(5)}, {[G, U, Q], WM(1)}, {[G, U], LR(5)}, {[G, U], WM(1)}, {[G], LR(5)}, and {[G], WM(1)}. It should be noticed that the number of predictors should be kept small to achieve acceptable performance. A maximum predictor number of 8 would be appropriate in practice.

DAS-2 clusters have a wide variety of users and different kinds of applications. In contrast to the EDG cluster, ex-

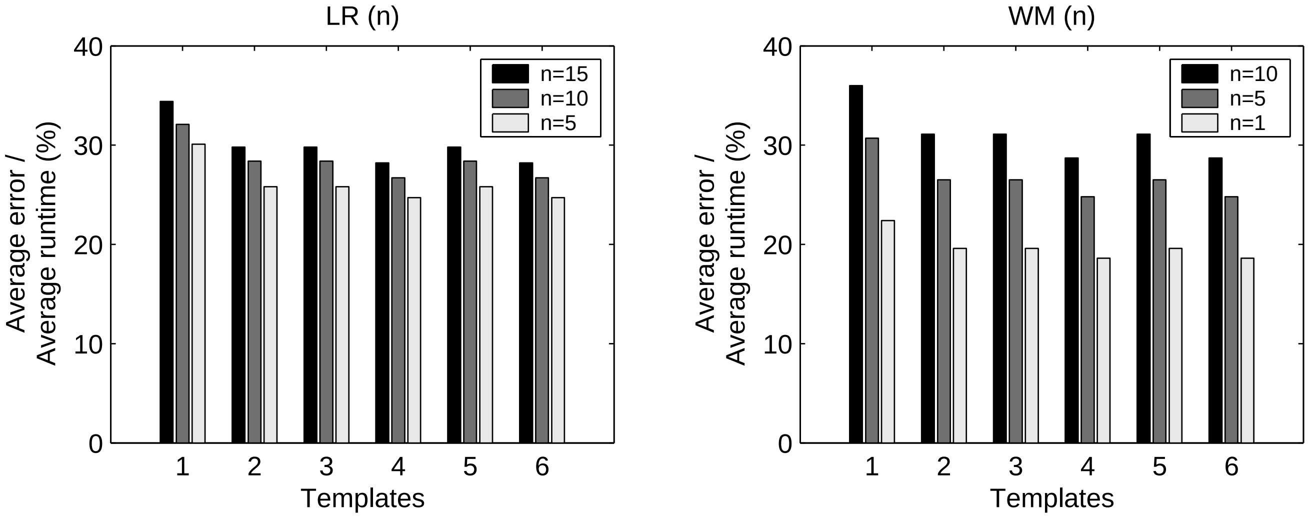

Figure 1. Quantitative analysis of templates and estimators for workload traces on the NIKHEF EDG production cluster. The average prediction error is defined as Average |estimated runtime - actual runtime|. The average job run time is 4672 seconds. Template labels are: 1 - [G] 2 - [G, U] 3 - [G, U, E] 4 - [G, U, Q] 5 - [G, U, E, N] 6 - [G, U, Q, E, N]

ecutable name (E) and number of processors allocated (N) prove to be useful information for categorizing jobs (see Figure 2 and 3). We conduct the same evaluation process as on the EDG cluster and find a suitable set of predictors on the DAS-2 clusters: {[G, U, E, N], LR(5)}, {[G, U, E, N], WM(1)}, {[G, U, E], LR(5)}, {[G, U, E], WM(1)}, {[G, U], LR(5)}, {[G, U], WM(1)}, {[G], LR(5)}, and {[G], WM(1)}.

2.3. Prediction generation

Instead of simply choosing the predictor with the smallest prediction error from the quantitative evaluation, namely {[G, U, Q], WM(1)} on the EDG cluster or {[G, U, E, N], WM(1)} on DAS-2 clusters, we introduce several techniques to generate predictions and compare their accuracy as well as performance.

The first technique we devise (referred as "LSTERR") is to dynamically select the predictor based on the previous prediction errors. We use *Least-Mean-Square* (LMS) to measure previous prediction errors

$$\xi_j = E[e^2{}_j(n)] = \frac{\sum_{i=1}^{n}(P_j(i) - D_j(i))^2}{n}, \qquad (1)$$

where $P_j(i)$ is the ith estimated job run time of predictor j, $D_j(i)$ is the ith actual job run time of predictor j, and n is the number of previous job entries used. ξ_j measures the average squared error of previous n predictions for predictor j. The predictor δ with the smallest ξ is selected to predict the run time of newly-submitted job

$$P(n+1) = P_\delta(n+1),\ min(\xi_j)\mid_{\,j=\delta},\ j \in [1, m], \qquad (2)$$

where m is the number of selected predictors.

The second technique (referred as "AVER") is to set the average of estimations produced by the selected predictors as the job run time prediction

$$P(n) = E[P_j(n)] = \frac{\sum_{j=1}^{m} P_j(n)}{m}, \qquad (3)$$

where m is the number of selected predictors.

We compare these two techniques with the predictor with the smallest prediction error (referred as "TEMP-EST") in the quantitative evaluation ({[G, U, Q], WM(1)} on the EDG cluster and {[G, U, E, N], WM(1)} on DAS-2 clusters). As can be seen in Figure 4, AVER has the smallest average prediction error and TEMP-EST performs the best on all three clusters. Since the performance of job run time predictions is not critical compared to the scheduler simulation in our system (to be discussed in Section 3), we take AVER as the overall best job run time predictor.

The basic algorithm of our job run time prediction technique is summarized as follows:

1. Define a set of templates and estimators based on the statistical properties of workload traces on the site.

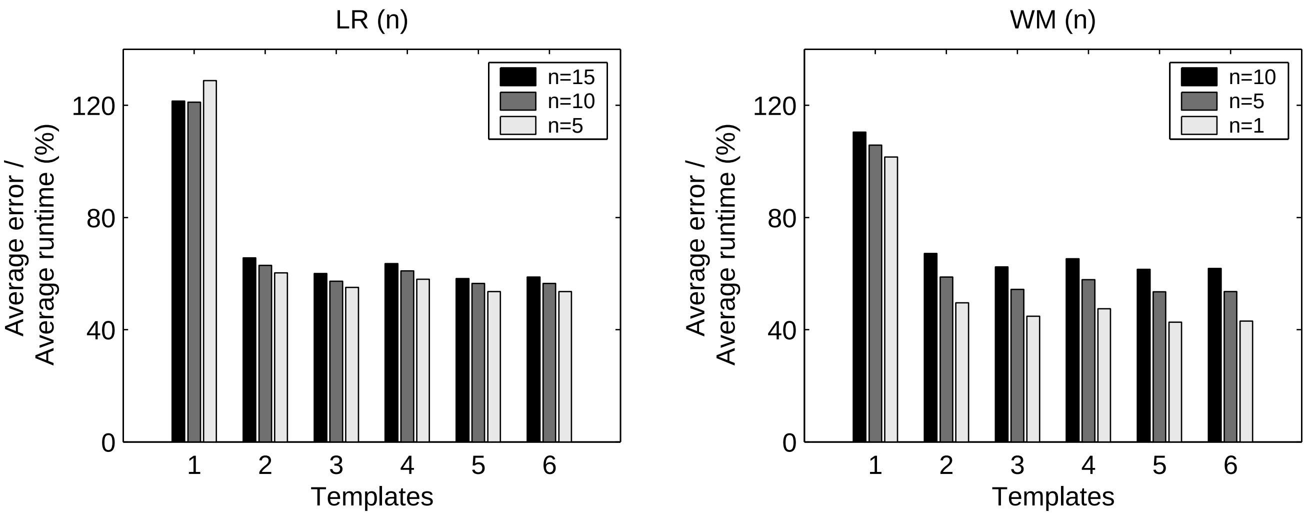

Figure 2. Quantitative analysis of templates and estimators for workload traces on the DAS-2 cluster at VU. The average prediction error is defined as Average |estimated runtime - actual runtime|. The average job run time is 524.2 seconds. Template labels are: 1 - [G] 2 - [G, U] 3 - [G, U, E] 4 - [G, U, Q] 5 - [G, U, E, N] 6 - [G, U, Q, E, N]

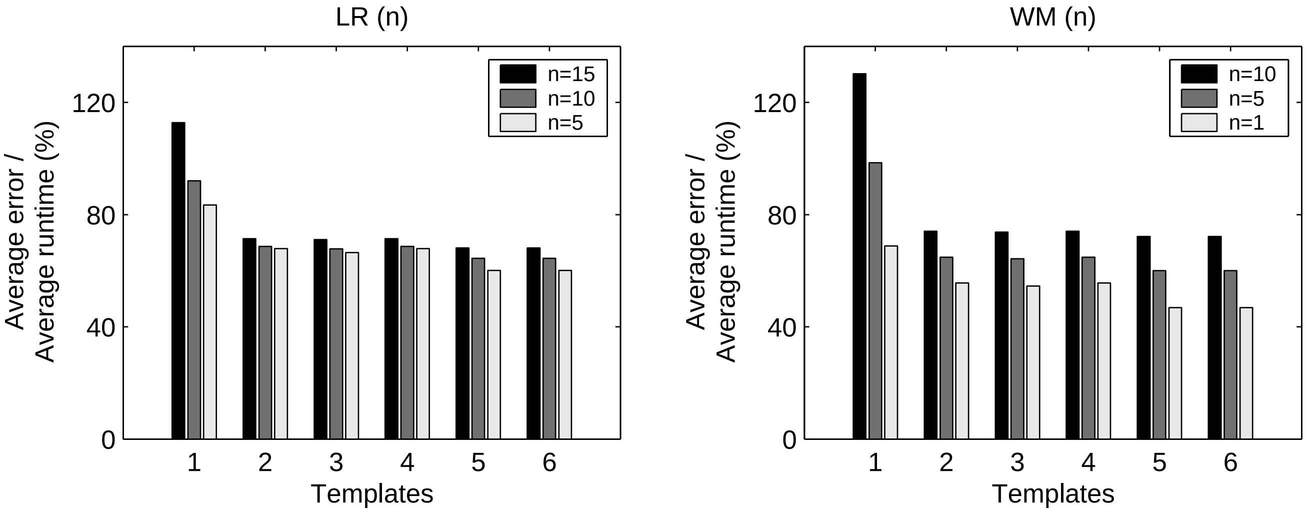

Figure 3. Quantitative analysis of templates and models for workload traces on the DAS-2 cluster at UvA. The average prediction error is defined as Average |estimated runtime - actual runtime|. The average job run time is 471.3 seconds. Template labels are: 1 - [G] 2 - [G, U] 3 - [G, U, E] 4 - [G, U, Q] 5 - [G, U, E, N] 6 - [G, U, Q, E, N]

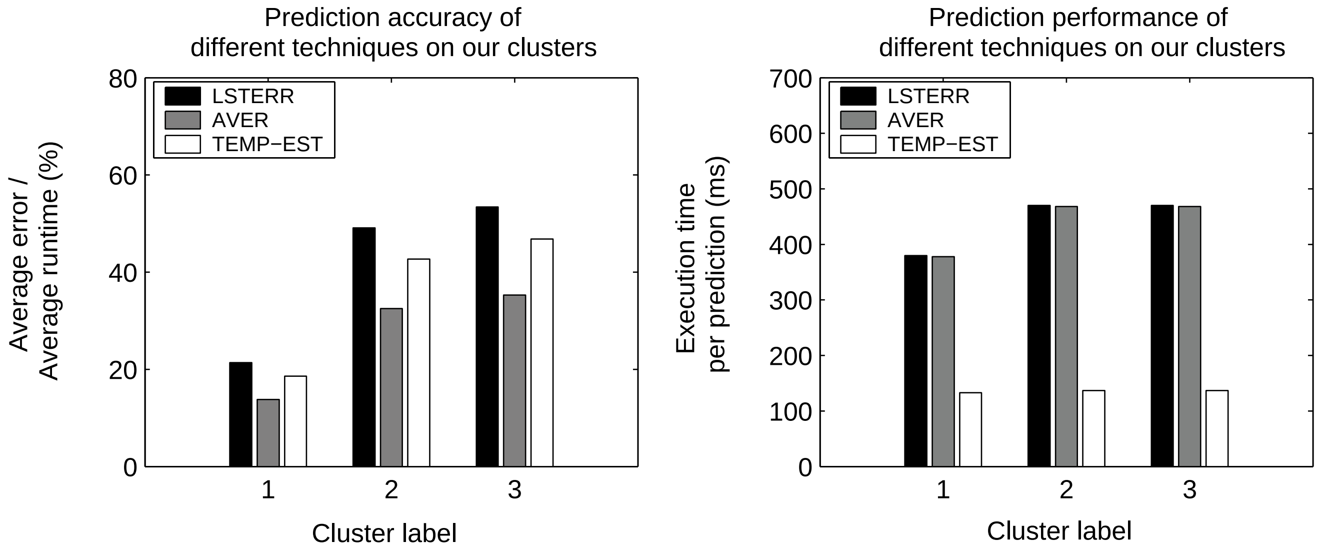

Figure 4. Comparison of three prediction generation techniques on our clusters. Average prediction error is defined as Average |estimated runtime - actual runtime|. The cluster labels are: 1 - NIKHEF EDG production cluster, 2 - DAS-2 cluster at VU, 3 - DAS-2 cluster at UvA (Execution times are measured on a 1 GHZ Intel PIII PC with 1 Gbyte internal memory).

Clusters	Average error of user specified wall clock times (s)	Average error of our predictions-AVER (s)	Average job run times (s)	Average error of our predictions/Average job run times(%)
NIKHEF EDG	150100	644.7	4672	13.8
DAS-2 VU	11960	170.4	524.2	32.5
DAS-2 UvA	3318	166.4	471.3	35.3

Table 2. Comparison of predictions based on user specified wall clock times versus our predictions.

2. Quantitatively evaluate the templates and estimators using historical traces and select a proper set of predictors (a predictor is an estimator combined with a template).

3. Evaluate different techniques (LSTERR, AVER and TEMP-EST) for prediction generation and choose the one with the smallest prediction error to implement the job run time predictor.

Table 2 shows the quantitative comparison of predictions based on user specified wall clock times versus our predictions. We can see that our predictions significantly reduce the average errors (up to magnitude of 2). The reason why predictions based on user specified wall clock times have such large errors is that users usually make very generous requests of wall clock times or simply do not specify them, in which case default wall clock times are applied. Our technique performs well, with average errors of 13.8, 32.5 and 35.3 percent of average job run times on the NIKHEF EDG cluster, the DAS-2 cluster at VU and at UvA, respectively.

3. Simulating the scheduling system

The second step to obtain job start times is via simulation. We snapshot the queue status on the site and simulate how the scheduler will schedule jobs. Since the actual execution times of running and queued jobs are unknown, we use our job run time predictions in the simulation.

In [12], Smith simulates First-Come First-Serve scheduling to predict job start times on clusters. This is a reasonable solution on one targeted site or sites with the same scheduling systems. However, in a Grid environment, different sites have different scheduling systems as well as site-specific policies. The simulator should be able to incorporate most of the popular scheduling algorithms and be easy to customize. Moreover, the simulation should achieve acceptable performance to fulfill a close-to-real time requirement. In this section we describe our approach towards scheduler simulation.

3.1. Maui simulation engine

The Maui scheduler [13] is a *policy engine* for clusters. It supports a variety of scheduling algorithms (FCFS, backfilling ,etc) and allows sites to implement various policies, partition resources and make reservations. Most EDG sites adopt Maui as scheduler, therefore simulating Maui becomes one of our main issues. Moreover, Maui is a super set of existing scheduling algorithms and we could easily mimic the behavior of other schedulers by customizing Maui. This makes our scheduler simulation more general and easily deployable to other clusters, even if they use different kind of schedulers.

Maui has a simulation mode for testing what the scheduler is capable of. It takes a resource file and a workload trace file and simulates how the scheduler will schedule jobs according to the Maui configuration file, in which site policies are defined. We could use this mode as our engine for simulation, but its performance becomes a major issue since it is not event-driven. On the NIKHEF EDG cluster, for instance, the simulation time versus the real time is roughly 1 : 90. Given this performance, if there are hundreds of jobs in the queue which take several days to finish, the simulation will run for hours. Therefore it is necessary to improve the simulation performance if we want to use this simulation mode for close-to-real time predictions.

3.2. Improving the simulation performance

The original Maui simulation mode has not been made event-driven because of the complexity that arises from dynamic policies. Our way to improve performance is to make the Maui simulation event-driven by controlling it outside. We also assume that the targeted sites do not employ dynamic policies such as dynamic prioritization, time based reservation, usage based throttling policies, etc.

The basic idea of our approach is to make the simulation “jump” from event to event instead of going through the whole simulation time. We take the basic events such as job start and end into account. With our job run time predictions we have the estimates when the running jobs will finish and we put their IDs as well as remaining run times into a run event list. We also maintain a queue event list for all queued jobs with the corresponding IDs and run time estimates in it. Firstly the run event list is examined and the smallest remaining run time is deducted from all jobs in the list. This makes the job with the earliest estimated finish time will terminate for sure next time when the simulation starts. We then construct a new workload trace file with jobs in both lists, start the Maui simulator with it, and advance the simulation for one interval (*interval* is a predefined value in seconds, each time we advance the simulation one interval, we advance the simulation clock this many seconds). Job(s) end and new jobs may be started. We remove finished jobs from the run event list, move newly started jobs from queue event list to the run event list and then stop the Maui simulator. We keep repeating the above steps until all jobs are finished. Through this process we speed the simulation up by making it event-driven outside the simulator engine. For details about the algorithm and implementation we refer to [15].

Cluster	Period	Average Maui simulation time(s)	Average event driven simulation time (s)
NIKHEF EDG	05-06/2003	1078	33

Table 3. Performance comparison of original and event driven Maui simulation.

Table 3 shows the performance comparison of original Maui simulation versus our event driven simulation on the NIKHEF EDG cluster. We can see that by making the Maui simulation event driven we improve the simulation performance to fulfill a close-to-real time requirement. This performance is sufficient, given the characteristic update time of 120 seconds in the EDG Information Service.

4. Start time prediction

The output of scheduler simulation is the start time predictions for queued jobs. However, these results are not directly usable and we have to publish the predictions in a way that a generic resource broker can make use of them. Our way of publishing predictions is based on site scheduling policies, which is discussed in this section. System design and experimental results are also presented.

4.1. Policy-based publishing

The output of scheduler simulation can be interpreted in different ways. We can publish the latest job finish time, job start times by queue or job start times by users. To incorporate site-specific scheduling information, we decide to publish start time predictions for groups (VOs) and/or users with policies defined. In practice, multiple predictions can be obtained within one simulation run. This is achieved by injecting probe jobs of targeted groups and/or users at the end of the current queue before starting the simulation. As soon as one of the probe jobs is actually scheduled, this event is recorded as the job start time prediction for the group/user it represents. When all probe jobs are successfully scheduled, we have the predictions for those entities with policies defined.

The NIKHEF EDG cluster can serve as an example to illustrate policy-based publishing. The policies for EDG jobs at NIKHEF are implemented as fair shares and limits, which can be specified in the Maui configuration file. Sample policies are:

```
GROUPCFG[default] PRIORITY=100
GROUPCFG[atlas] PRIORITY=300 MAXPROC=4
GROUPCFG[alice] PRIORITY=500 MAXPROC=6
USERCFG[svens] PRIORITY=100 MAXPROC=2
```

Policies can be defined for groups and/or users. For instance, the priority of jobs from group *atlas* is set to “300” and the maximum processors that can be used by this group is “4”. Because of these different policies, different groups and users would most likely have different job start times. Here is a snapshot of site queue status:

```
ACTIVE JOBS--------------------
JID USER STATE PROC REMAIN STARTTIME
31717 svens Running 1 2:20:30:28 ...
31718 svens Running 1 3:14:37:09 ...
31815 atlas001 Running 1 3:22:00:10 ...

4 Active Jobs, 4 of 20 CPUs Active(20%)
IDLE JOBS--------------------
JID USER STATE PROC REMAIN STARTTIME
31820 svens Idle 1 4:00:00:00 ...
31821 svens Idle 1 4:00:00:00 ...
31823 svens Idle 1 4:00:00:00 ...
31832 svens Idle 1 4:59:59 ...

Total Jobs: 8 Active: 4 Idle: 4
```

We can see that although there are many free processors available at the site, jobs from user *svens* still have to wait in the queue since policies restrict the maximum number of processors that can be used by him is 2. At that time, jobs from *svens* will surely have larger start times than other groups and users. The results of our job start time predictions are:

```
default: 0 (Mon May 12 15:20:28 2003)
atlas: 0 (Mon May 12 15:20:28 2003)
alice: 0 (Mon May 12 15:20:28 2003)
svens: 82605 (Tue May 13 14:16:58 2003)
```

It shows that different groups and users do have different job start times, according to the site policies defined. The resource broker can match appropriate job start time predictions to decide where to submit jobs.

4.2. System design

The design of the job start time prediction system is illustrated in Figure 5. The system consists of four main components: *a historical database*, *a queue monitor*, *a job run time predictor* and *a start time predictor*. The historical database stores the categorized historical job entries, which are used for job run time predictions. The queue monitor gets the real-time queue information on the site and provides it to other components. In the start time predictor, the controller controls the Maui simulation engine to perform event-driven simulations. Job run times used in the simulation are obtained through the job run time predictor in real time. The start time predictor publishes the prediction results according to the scheduling policies defined at the site.

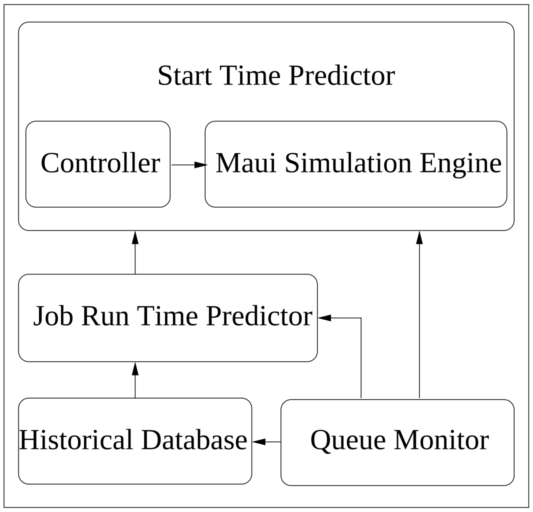

Figure 5. Design of the start time prediction system

Currently the system runs on sites with PBS [14] as batch system and Maui as scheduler. For implementation details we refer to [15].

4.3. Experimental results

We have deployed the job start time prediction system on the NIKHEF EDG production cluster. As is shown in Table 4, the average error of our predictions is 18.9 percent of the average queue wait time and it is around 20 times smaller than the average error of the EDG solution. We also studied start time predictions for different VOs and users.

Cluster	Period	Average error of EDG predictions (s)	Average error of our predictions (s)	Average queue wait time (s)	Average error of our predictions /Average queue wait time (%)
NIKHEF EDG	05-06/2003	325144	15276	81061	18.9

Table 4. Comparison of EDG start time predictions versus our start time predictions

The average prediction errors vary from 8 to 200 percent of the average queue wait times [15]. Some users always submit similar jobs (e.g., a group of physicists who are analyzing data from a large High Energy Physics experiment). They have very predictable job run times and the average start time prediction error is small. Some users (e.g., software testers and integration team) submit jobs which vary a lot in execution times and they have relatively large prediction errors. Generally speaking, our job start time predictions significantly improve the originally implemented EDG solution.

5. Conclusions and future work

In this paper we present a job start time prediction system for computer clusters. Job start time predictions can be used by middleware component such as a resource broker to balance the workload distribution. It can also be used to compute “resource price” in a grid accounting system [16]. In our system, we use historical information and statistical techniques to predict job run times, and make scheduler simulation to obtain job start times. We publish job start times by group and/or user according to the site policies. We have deployed the system on NIKHEF EDG production cluster and find that our predictions achieve acceptable accuracy to reflect the real site states.

Our system has its limitations as well. It runs slower than the EDG solution because of the statistical predictions and simulations involved. Although we have made the simulation event-driven, the simulation time grows linearly as the number of queued jobs increases. Also, the system assumes that the site does not employ dynamic scheduling policies.

In future work, we aim to further improve the system performance (e.g. by extrapolating from jobs in the same category that are already queued). Proper schema will be created in order to publish multiple job start time predictions to the Grid Information Service. We also take interest in further studying statistical properties of EDG workloads and evaluate more estimators for run time predictions.

6. Acknowledgments

The EU DataGrid project is funded by the European Commission under contract IST-2000-25182. This work is part of the research program of *Foundation for Fundamental Research on Matter* (FOM) , which is financially supported by *Netherlands Organization for Scientific Research* (NWO). We want to express our gratitude to Warren Smith (NASA Ames), Thomas Roblitz (ZIB) and Rosario Piro (INFN) for their insightful discussions and valuable suggestions. We’d also like to thank our colleagues at NIKHEF: Martijn Steenbakkers, Oscar Koeroo and Kors Bos.

References

[1] European Union DataGrid. http://www.eu-datagrid.org/.

[2] Definition of architecture, technical plan and evaluation criteria for scheduling, resource management, security and job description, DataGrid-01-D1.2-0112-0-3, 2001.

[3] Information and monitoring service architecture, DataGrid WP3 draft document, 2003. http://hepunx.rl.ac.uk/edg/wp3/.

[4] Richard Gibbons. A Historical Application Profiler for Use by Parallel Schedulers. *Lecture Notes on Computer Science*, Vol. 1297, pages 58-75, 1997.

[5] Allen B. Downey. Predicting Queue Times on Space-Sharing Parallel Computers. In *Proceedings of the 11th International Parallel Processing Symposium*, pages 209-218, 1997.

[6] Warren Smith, Ian Foster and Valerie Taylor. Predicting Application Run Times Using Historical Information. *Lecture Notes on Computer Science*, Vol. 1459, pages 122-142, 1998.

[7] NIKHEF EDG testbed. http://www.nikhef.nl/grid.

[8] DAS-2: the Distributed ASCI Supercomputer - 2. http://www.cs.vu.nl/das2/.

[9] P.A. Dinda and D. O’Hallaron. An evaluation of linear models for host load prediction. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing(HPDC’99)*, 1999.

[10] Erwin Kreyszig. Introductory Mathematical Statistics - Principles and Methods. John Wiley and Sons, Inc. ISBN 471 50730 X, 1970.

[11] N.R. Draper and H. Smith. Applied Regression Analysis, 2nd Edition. John Wiley and Sons, Inc. ISBN 0-471-02995-5, 1981.

[12] Warren Smith and Parkson Wong. Resource Selection Using Execution and Queue Wait Time Predictions. Tech. Rep. NAS-02-003, NASA Ames Research Center, July 2002.

[13] Maui scheduler. http://www.supercluster.org.

[14] Portable Batch System (PBS). http://www.openpbs.org.

[15] Hui Li. Master’s Thesis. *Predicting Job Start Times on Clusters.* Internal Report 03-11, Leiden Institute of Advanced Computer Science, Leiden University, 2003.

[16] DataGrid Accounting System. http://www.to.infn.it/grid/ accounting/.