# Predictive Resource Scheduling in Computational Grids*

Clovis Chapman, Mirco Musolesi, Wolfgang Emmerich and Cecilia Mascolo
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, United Kingdom
{c.chapman, m.musolesi, w.emmerich, c.mascolo}@cs.ucl.ac.uk

## Abstract

*The integration of clusters of computers into computational grids has recently gained the attention of many computational scientists. While considerable progress has been made in building middleware and workflow tools that facilitate the sharing of compute resources, little attention has been paid to grid scheduling and load balancing techniques to reduce job waiting time. Based on a detailed analysis of usage characteristics of an existing grid that involves a large CPU cluster, we observe that grid scheduling decisions can be significantly improved if the characteristics of current usage patterns are understood and extrapolated into the future. The paper describes an architecture and an implementation for a predictive grid scheduling framework which relies on Kalman filter theory to predict future CPU resource utilisation. By way of replicated experiments we demonstrate that the prediction achieves a precision within 15-20% of the utilisation later observed and can significantly improve scheduling quality, compared to approaches that only take into account current load indicators.*

## 1 Introduction

During the last five years, computational scientists have started to adopt Grid computing techniques and infrastructures in earnest. This has, in part, been enabled by the increased computational power and available capacity from commodity equipment, and by the emergence of inter-organisational, national and international grid computing infrastructures, such as the e-Minerals Grid [4] in the UK, the Tera Grid [1] in the US or international grids, such as EGEE [10].

*The research described in this paper has been partially funded by the UK DTI through grant THB/008/00175C (WS-Condor), the UK NERC through Grant RG33981 (*e*Minerals) and the UK EPSRC through grant EP/C544765/1 (CREAM).


The vision of *grid computing* [2] is to integrate clusters into global infrastructures in such a manner that users no longer need to be aware of which computational resources are used for executing their jobs and storing their data. This requires solutions to a number of problems, including authentication and authorisation, reliable file transfer, distributed storage management and resource scheduling across organisational boundaries, which is the focus of this paper.

In order to achieve such a level of integration, the question emerges of which cluster should be used to solve a particular computational task. Obviously, the clusters have to match the resource requirements of the jobs at hand, but apart from that, there may well be a significant degree of freedom as to where jobs should be executed. Initially, it was the scientists who had to make that decision and this was often based on resources they knew about and had access to. Once a cluster was identified the scientists submitted a job, typically a batch process, to a *distributed resource management system* (DRM), such as Condor [21], PBS [13], or the Sun Grid Engine [11]. A DRM is responsible for allocating the job to a node using some resource allocation policy that may take into account node availability, user priorities, job waiting time etc. Alternatively, the job could be submitted to a service that abstracted away from proprietary resource managers, such as the Globus GRAM [7] or the GridSAM web service [16].

Today, those decisions are made, in an automated manner, by portals, meta-schedulers or even through a federation of resource managers. Meta-schedulers perform top-down scheduling decisions and federation of clusters that perform load balancing between DRMs in a peer-to-peer manner. Examples of such meta-schedulers include Condor-G [21] and compute portals [22], both of which support the scheduling of jobs across computational grids. Examples of federation technologies include the flocking techniques available in Condor [21].

The fundamental problem that all of the above techniques have to solve is to select a DRM that can then schedule a job on the resources that it controls. The problem

is made more difficult due to the level of autonomy of DRMs that in general accept job submissions from more than one meta-scheduler in addition to submissions from local user communities. This level of autonomy prohibits a centralised solution to meta-scheduling. Users are interested in completing their jobs as quickly as possible. When optimising selection in this respect, different strategies can be employed: the most commonly used one is to base the decision on current utilisation, measured, for example, by queue length in relation to the number of resources available. This approach delivers suboptimal results whenever there is a significant standard deviation in the average job length per cluster. In those circumstances it may well be advantageous to submit a job to the DRM controlling the cluster that appears to have the higher utilisation.

In this paper, we argue that scheduling based on future resource utilisation improves the user's quality of service. However, this requires the ability to predict future utilisation on clusters. The main contributions of our work are the design and evaluation of a predictive scheduling approach for resource utilisation based on linear Kalman filters, a state-space model forecasting technique. Focusing primarily on CPU utilisation, our approach exploits the fact that users' job submission patterns are repetitive. To prove this point, we have evaluated the utilisation history of a 940 node cluster, which has been used by approximately 30 different scientists from a number of different disciplines over a period of two years. We found that the job submission patterns are highly repetitive in several respects, and based on these data show how linear Kalman filters are able to detect these patterns and predict the execution time of jobs within 15-20% of accuracy. We present an architecture for such predictive grid scheduling and discuss how it can be implemented using the Condor resource manager. Through replicated experiments with utilisation observed in practice, we demonstrate that allocation decisions based on these Kalman filter predictions can reduce waiting time compared to naïve scheduling based on shortest queue lengths, while the computational costs for using Kalman filter scheduling are negligible.

The paper is further organised as follows: Section 2 contains the description of our approach and illustrates the architecture and implementation of our framework. In Section 3, we present the results of our experimental evaluation and describe the advantages of the approach. Section 4 contains a discussion of related work, while Section 5 concludes the paper by illustrating possible future directions.

## 2 Predictive scheduling

In 2002, we have been instrumental in establishing a large cluster at UCL. The cluster uses Condor to scavenge otherwise unused CPU time of some 1,100 student

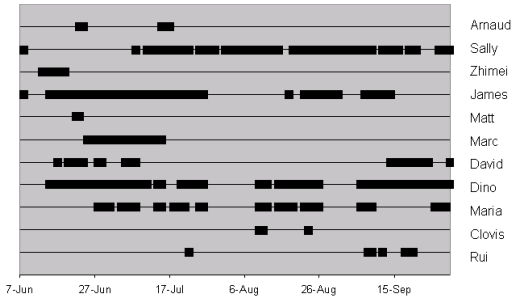


**Figure 1. User activity during analysis period.**

Windows workstations to process chemistry, physics and geological computations, in the form of batch processes. Largest of its kind in the UK, the Condor cluster is being actively used by some 30 computational scientists.

We have retained utilisation logs of the UCL Condor cluster for two years. The volume of computation is such that we argue that it is comparable to a dozen clusters of average size and it therefore represents a good sample. Following high level screening, we have selected a period of four months for detailed analysis, characterised by a higher than usual level of activity.

Rather than observing jobs anonymously, our detailed analysis investigated the time and number of jobs submitted, as well as the length of computation, on a user-by-user basis. That analysis revealed a number of very interesting results. Figure 1 shows the times when the most active eleven users submitted jobs to the cluster during the four month period that we analysed in detail. A rectangle against a user and a date indicates that the user has submitted jobs during that day. The key observation to be derived from that figure is that, even though these were the most active users, they did not all submit jobs all the time. Moreover, there are very detectable patterns of job submission. For example James and Dino use the cluster for extended periods of time. Arnaud and Maria use the cluster for 3-6 days of intensive activity followed by a period of inactivity that is at least equally long.

Figure 2 shows the distribution of job arrivals during a randomly selected three day period. The figure shows three data series, one for each day, where each column shows the number of jobs that were submitted in any six minute interval. What can be seen from Figure 2 is that, on each day, job submissions do not occur with a regular distribution but users submit large numbers of jobs in one go and then there are extended periods where no jobs are submitted. The reason for this behaviour is that users rarely submit jobs manually but, rather, automate submission using computational workflow tools, such as the OMII-BPEL environment [9], Condor's DAGMan [21] or simply with shell scripts.

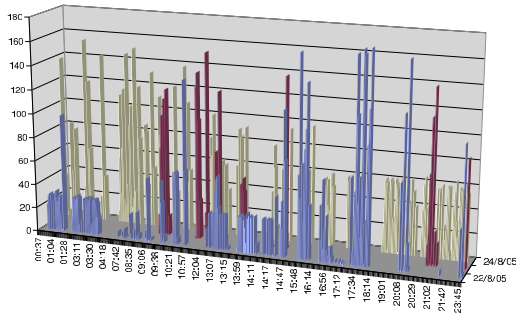

**Figure 2. Job submission rates during three randomly selected days.**

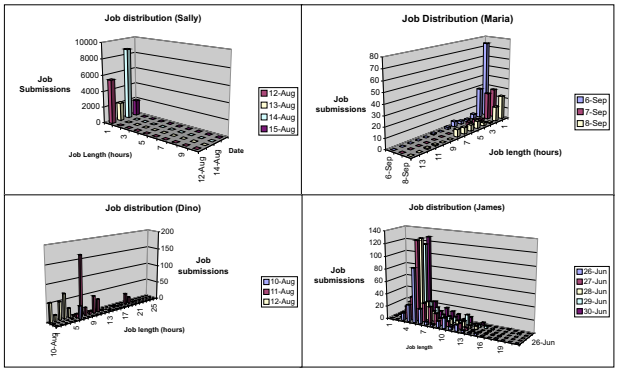


**Figure 3. Job length by users.**

The most interesting observation is shown in Figure 3. The figure shows the distribution of job lengths for the four most active users. It shows that the vast majority of jobs that Sally submits are shorter than an hour, while Dino regularly submits long running jobs the length of which often exceeds 20 hours. James submits jobs with an average length of about 7 hours and his job distribution has a non-negligible standard deviation. The root cause for these observations is that users utilise the same set of computational applications over extended periods of time, within or across sessions, and apply them to different studies and data sets.

The discussion of the empirical samples above shows that the behaviour of users who submit jobs to a grid is not at all random, but, rather, follows regular patterns. It is this observation that we use and exploit in our aim to improve the quality of scheduling decisions that meta-schedulers, portals or federations of grids need to make. The basic idea is that, because job submission occurs in distinctive patterns and the duration of jobs may vary between minutes and days, established prediction techniques may yield better scheduling decisions than the standard technique of submitting to the DRM, which has the cluster with the shortest queue length or the lowest current load. We use this insight to derive the two main hypotheses of this paper.

Our first hypothesis is that we can use the job execution history and resource utilisation of the past to predict resource utilisation and hence availability in the immediate future linear Kalman Filter theory. The second hypothesis of this paper is that we can improve the quality of grid scheduling in comparison to techniques that use shortest queue length, by using this ability to forecast resource utilisation.

## 2.1 Prediction Theory

Users are generally interested in minimising the time between their submission of a job and its completion. We refer to the elapsed time between these events from the point of view of a DRM as the *job turn-around time*. The utilisation of the cluster by other users may affect the level of service obtained and, as a consequence, the overall length of a job, as the various users compete for underlying resources.

We predict future cluster utilisation by applying Kalman filter forecasting [15], originally developed in automatic control systems theory and applied in many different fields, from telecommunications to weather forecasting. Kalman filters are essentially a method of discrete signal processing that provide optimal estimates of the current state of a dynamic system described by a *state vector*. The state is updated using periodic observations of the system, using a set of *recursive prediction equations*. In this manner, we can calculate, at time $t$ the predicted value of the utilisation time $t + h * T$, with $h > 0$ (i.e., after $h$ sample intervals). A detailed mathematical description of the model we have relied upon is presented in [17].

In a grid environment, numerous indicators of utilisation can be selected as a basis for these observations, such as job queue lengths and resource availability, the choice of which will depend on resource management systems and scheduling policies in place and their effect on the duration of the lifetime of a job. We discuss the choice of utilisation factors and job length indicators in Section 2.2 and our specific choice of indicator in our implementation in Section 2.3.

One of the main advantages of Kalman filters is that there is very little computational and storage overhead as they are expressed through recursive equations. The entire history of the system does not have to be maintained and it is sufficient to record the value of the current inner-state and the parameters of the recursive equations, updated at every step. This low amount of state ensures better scalability in large scale grid environments. Moreover, this class of predictors does not require a training phase, unlike other types of forecasting techniques based on machine learning [18]. However, an initial set of parameters used to bootstrap the system must be tuned to the environment and indicators at

hand. This is further discussed in Section 2.3.

## 2.2 Architecture

In this subsection we explore how the Kalman filter based prediction technique can be used in an architecture of production level grid environments to improve cross-organisational scheduling decisions. Key components that

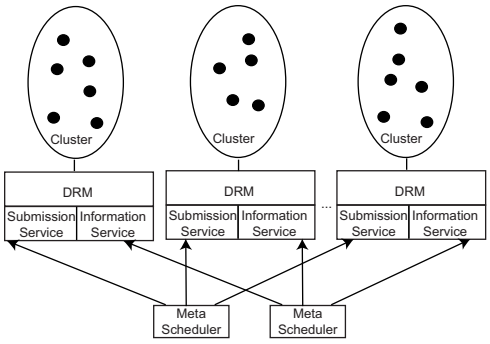


**Figure 4. Architecture of Grid environment**

are of interest to us here are illustrated in Figure 4:

- *Distributed Resource Management system:* As previously stated, the resource management system is responsible for managing a collection of resources at each site, allocating resources to tasks, according to policies determined by the resource owners.
- *Submission services:* These components provide a service abstraction for job scheduling and resource allocation at each site offering means for clients to delegate the responsibility of managing and allocating resources in the local environment to a scheduler or job queue local to the site. Jobs are submitted to these services in the form of a job description and associated input files and executables.
- *Information services:* Such services will provide descriptions of underlying resources, such as availability and characteristics of the resources available at a site. These enable remote clients to determine the suitability of a resource and the potential utilisation they may obtain from a site before submitting their job for execution.
- *Meta-schedulers:* Meta-schedulers are responsible for managing the use of resources at multiple sites on behalf of a user. They are responsible for the process of resource selection and job distribution across sites. Meta-schedulers can take various forms, from web-based portals providing an interface through which a user can submit jobs, to local applications responsible for generating and managing jobs according to a computational workflow specified by the user.

It is the meta-scheduling process that we are specifically concerned with here. Faced with a potentially large number of sites providing resources of comparable capabilities and matching the job requirements, careful selection is required to maximise the throughput of jobs submitted by the user. Relying solely on information currently made available by the resource information services is not sufficient to determine the suitability of a particular site and we must also avoid potential starvation of the meta-schedulers awaiting resources. Taking into account predicted utilisation patterns can considerably improve the meta-scheduling process.

By making more informed decisions as to where to submit the job, we can reduce the waiting time and global scheduling overhead of the job and as a consequence increase the throughput of the meta-scheduler.

The attributes that will affect the duration of the lifetime of a job at a site between submission and completion can vary according to scheduling policies in place and the process by which resources are allocated to competing users and tasks. Beyond the capacity of the individual resources, which will directly affect the length of the execution of a job, other job length indicators can be considered. For example, in the context of a simple batch queuing system operating on a FIFO basis, such as the Portable Batch System, we can potentially rely on the number of jobs in the queue to be a direct factor of how long a specific job will have to wait before being served. However, this figure may be misleading if we do not take into account past utilisation trends. Whilst a site may have a larger queue than another at any particular time, an analysis of utilisation trends using Kalman filter forecasting techniques might reveal that the queue length varies much more rapidly at one site and consequently that jobs queued are potentially shorter, making it a much more suitable choice for the meta-scheduler. On the other hand, Condor – as we will see in the following section – operates according to a policy-based resource allocation mechanism, where resource claims are a more suited choice of indicator.

The selection of job length indicators should be adapted to policies in place, but it may also be beneficial to consider other attributes of the environment, such as network bandwidth: jobs submitted may require or can produce considerable amounts of data, and the amount of time needed to transfer that data may have an impact on the overall job length.

Forecasting by using Kalman prediction requires regular readings of the state of the resources, or, more specifically, the selected job length indicators, to be made at the targeted cluster in regular intervals. For this purpose, it can be very valuable for the predictions to be built within the administrative domain of the site before being communicated to re-

mote meta-schedulers. A prediction service residing within the domain, may have access to finer grain utilisation data such as per user job activities and specific job type characteristics that a site may not be willing to share with an external observer. However, because the Kalman filters do not require the past history of the system, it is perfectly possible for an external observer, such as the meta-scheduler itself, to build predictions based on information obtained from the Resource Information services, around the period of time desired. Whilst coarser grain utilisation data may have to be relied upon, such as general resource availability, an external service operating on behalf of a user can restrict its observations to the specific subset of resources at a site that will match the job requirements of this user, and fine tune the parameters of the prediction according to the specification of the overall computational process. We also cannot assume that a prediction service will be available at every accessible site. The ability to conduct these predictions independently and with minimal input from site providers is an important characteristic of our approach in an environment where site autonomy and heterogeneity is the norm.

### 2.3 Condor Implementation

We have used the Condor job scheduling and resource management system and GridSAM to implement a meta-scheduling environment that relies on the above Kalman filter forecasting techniques and architecture to select suitable clusters of Condor resources on which to schedule jobs.

We build on previous work [12] that involved incorporating web service support into the Condor architecture, by exposing key functionality of Condor, such as resource information and job submission as individual web services. We have also, in the context of this work, created a Condor plugin for GridSAM. The GridSAM service enables users to remotely submit jobs to a wide range of underlying resource management systems in the form of Job Submission Description Language (JSDL) documents, an emerging GGF standard [16]. Jobs submitted to GridSAM are delegated to DRMs through a collection of DRM-specific plugins. By implementing a plug-in for GridSAM that relies on Condor's web service interface to remotely interact with Condor, we have provided it with the capability to schedule jobs across multiple sites and it is on this service that we rely here to act as a meta-scheduler for our system.

We can relate our implementation to the above architecture as follows:

- *Distributed Resource Management System:* The Condor resource management system is responsible for managing the site at the resource level.
- *Job Submission Service:* The Condor scheduler, for which there may be multiple in a single Condor cluster, is responsible for managing a queue of jobs on behalf of the user and managing their remote execution in the local cluster. It provides a Web Service interface for job submission and monitoring.
- *Information Services:* The Condor Collector is responsible for collecting and providing meta-data about the current state of resources in the cluster – which may be either static characteristics (e.g. OS, total memory, etc.) or dynamic characteristics such as current availability. Meta-data queries can be issued by remote meta-schedulers through the collector's Web Service interface.
- *Kalman Prediction service:* Alongside the above services, we have created an additional Kalman Prediction Service. This service is responsible for obtaining, periodically, meta-data from the resource information services, and use this meta-data to construct predictions as to future resource availability.
- *Meta-scheduler:* We rely on the GridSAM service and our Condor plug-in to provide meta-scheduling capabilities. We have extended GridSAM with the ability to query either resource information services or Kalman prediction services at one or more specified sites in order to determine the most suited location to which to submit an incoming job.

The scheduling and allocation process of Condor needs to be carefully studied in order to determine the factors that influence the turn-around time. Condor clearly separates the process of allocation and scheduling. Jobs submitted to a Condor scheduler are maintained in a user specific queue of jobs. During a regular negotiation cycle, the Condor central manager will allocate available resources to jobs enqueued by the various schedulers according to user priorities and other community policies - such as fair share policies. The scheduler, once allocated one or more resources, will *claim* these resources on behalf of the user, and maintain these claims until it no longer requires them. It may also request more resources from the central manager at regular intervals. In an environment where numerous users will be competing for available resources, and when no resources are immediately available, the waiting time will primarily be constituted of the estimated amount of time it will take for a resource to be freed or released by a previous owner, or the amount of time it may take for a user's jobs to complete on resources that have already been claimed by that user.

Condor does support much more complex policies that are not in use in our Condor environment, allowing for example preemption of claims and jobs in favour of higher priority users. In such cases, a new single or combination of job length indicators, adapted to the policies in place,

should be selected to perform the prediction. In this setting, we rely on claims as the primary indicator.

The prediction service will – on a specified time interval – read the current number of unclaimed resources and use that information to update the state space model of our predictor. This will provide an estimate of the number of unclaimed resources that will be available at the next time step. Where several clusters might appear to be full, the ability to predict the number of resources that will be freed in the next time interval, can make a considerable difference. We can, if needed, use the Kalman filter forecasting technique to forecast availability several steps beyond the specified time interval, by simply feeding predictions back into the predictor. Our current implementation of the service will enable meta-schedulers to specify the number of steps it may wish to consider in advance, though there is of course a trade off between accuracy of the prediction and the time frame that we take into account.

Other parameters that may affect the accuracy of the prediction, such as the periodic time interval, the amount of time that may be required to run the predictor before it identifies utilisation patterns accurately and the degree of correction when calculating the state of the current dynamic system of our predictor will also require fine tuning and experimentation to define an optimal level of accuracy suited to various workloads. In particular, the latter will affect the adaption of the time series of predicted values in presence of fluctuations, essentially defining how the filter should respond to strong variations in the readings. The setting of these values, alongside the initial value of the state, are the only steps required to bootstrap the filter.

As such, the meta-scheduling process works as follows: users will submit their jobs to a host running a GridSAM service. Upon receipt of the job description document, the GridSAM service will query all the Kalman prediction services at the sites specified by the user and obtain their latest prediction for resource availability. It will select the site that provides the highest number predicted of unclaimed resources in the next time interval, and submit accordingly.

It should be noted that it is not necessary for submissions to be restricted to specific time slots that match the periodic time interval used by the Kalman service to construct predictions, as the prediction does not just indicate the potential number of available resources in the near future, but also the general trend of the time series.

# 3 Evaluation

## 3.1 Experiment Design

To evaluate our Condor-based predictive meta-scheduling framework, we use an experimental evaluation technique and set up a replicated experiment that compares

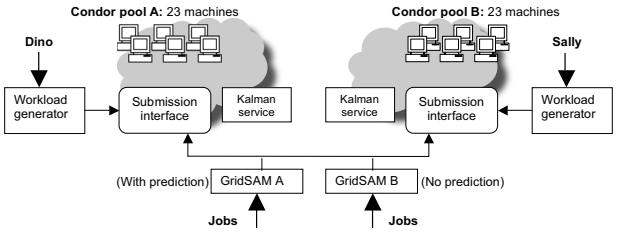


**Figure 5. Test-bed Architecture.**

predictive scheduling with scheduling based on current resource availability (i.e., current queue lengths).

Figure 5 shows an overview of the experimental test-bed. The test-bed comprises two Condor clusters of 23 nodes each. Whilst this is a considerably lower count than the UCL cluster of 1,100 nodes, these clusters are still average-sized (the grid described in [4] only has 16 nodes per cluster) and is sufficient for our evaluation.

We assume that each cluster has local users, which are simulated using a workload generator based on the utilisation patterns observed at the UCL Condor cluster that we have discussed above.

It is important to define our experimental space: Kalman prediction is of most benefit where clear and distinct patterns of use may emerge for each resource. As we have seen in Section 2, job submissions will present such patterns. In scenarios where multiple users submit independently, the combined collection of workloads will result in a new pattern. Based on the multiple experiments that we have conducted using various combinations of usage patterns observed in our pool, we have always obtained some level of improvement using prediction. However, where all resources present near identical patterns of use we can, of course, expect little or no improvement over the use of current resource availability, primarily as there would be no real benefit to choosing one resource over another: jobs submitted to either site would take on average an equal amount of time. The closer the usage patterns, the less gain there is to be had from using prediction.

In order to corroborate these claims, we detail two of the experiments we have performed that represent two extremes. The first one uses very different workloads that have been discussed in Section 2. We expect the Kalman filter prediction to perform very well in this experiment. The second one uses near identical workloads on both clusters and we expect to see no significant improvement but want to validate with the inclusion of this example that there are no disadvantages from using Kalman filter prediction.

For the first experiment, we have specifically chosen local workloads that mimic Sally and Dino due to the very different utilisation patterns that both present. By extrapolating job length distributions from our logs, the workload

generator mimics the patterns of these users using sleeper jobs aiming to occupy the resources for varying periods of time. Though the workloads have been adapted proportionally to the size of our new clusters, they will retain the same characteristics:

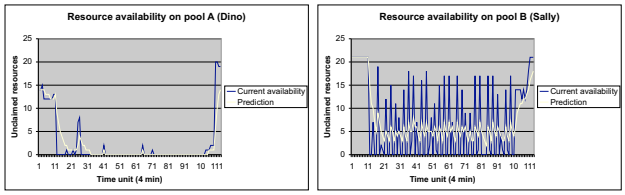


**Figure 6. Real vs. Predicted Availability.**

- *Sally:* Sally's workloads are submitted according to a workflow process, which will submit large collections of short jobs organised in various subsets. A description of the scientific objectives of the process and the computations actually performed is outside the scope of the paper – we refer the reader to [9]. It is sufficient to know that in a first stage a small collection of jobs will launch the process. The length of these may vary from a minute to several hours. Each completion will immediately trigger a further 200 submissions of jobs, which are characterised by very short execution times (between one and five minutes). The total process in a single session can reach a maximum of 7200 job submissions. The result is a pattern similar to what is illustrated on the right-hand side of Figure 6. For our purpose, we have only selected a particular subset of the workflow suited to our test-bed environment, with a more manageable number of simultaneous submissions (up to 30).

- *Dino:* In contrast, Dino will submit collections of long running jobs. Though the number of simultaneous submissions is considerably smaller than Sally's, the length of the jobs can exceed 20 hours. Upon completion of individual job instances, a new job might be submitted to further refine the produced data. The resulting pattern can be seen on the left-hand side of Figure 6. Again for the purpose of our experiment, we have selected a subset of Dino's workflow where the execution length is more manageable on our test-bed, with individual jobs last between 40 and 60 minutes.

## 3.2 Prediction Quality

Figure 6 shows a comparison between the real and predicted resource availability on the two clusters. Table 1 represents the overall level of accuracy of the prediction obtained at each site. As a starting value for the Kalman filter we use an initial reading of the state of the pool on beginning the experiment. Upon computing predictions, we also calculate the confidence interval of the predicted value. This interval can be used as an early indicator of the validity of the prediction.

The actual level of accuracy that we obtained can be represented by the standard deviation of the error of the prediction. As expected, the accuracy of the prediction was greater when dealing with Dino's workload than Sally's due to the high variability of the latter. The average of the error is around 0 in both cases, implying that the filter is not over nor under-estimating the future utilisation of the clusters.

| Kalman Predictor | on Cluster A | on Cluster B |
|---|---|---|
| Periodic time interval [mn] | 4 | 4 |
| Avg. prediction error | 0.0206 | -0.02511 |
| Std. deviation | 2.922 | 5.5 |
| Avg. confidence interval | 0.607 | 0.607 |

**Table 1. Scheduling Quality.**

## 3.3 Improvement of Scheduling Quality

In order to demonstrate that knowledge of future availability can improve meta-scheduling performance considerably compared to using current utilisation indicators, we have set up two GridSAM nodes which will submit jobs on a regular time interval to the clusters. One of these nodes, GridSAM node A, in Figure 5, will rely on information obtained from the Kalman prediction service at each site to determine which resource will have lower future utilisation. In contrast, GridSAM node B will base its selection solely on current utilisation data obtained from the resource information services at each site.

The jobs submitted by GridSAM nodes A and B, have the sole purpose to allow us to measure the amount of time they will remain enqueued before a resource is freed. In order to ensure that A and B encounter the same exact environment when scheduling these jobs, these will be made to operate simultaneously and with the same time intervals for job submission. These jobs will run exactly for 10 seconds, once allocated a resource and their overall impact on the site workloads will be negligible. For a hundred submissions, the time of submission and completion of a GridSAM job will be recorded, and these figures will then be used to determine which meta-scheduler provided us with the highest overall throughput.

We do make several assumptions as to the nature of the environment and jobs. First of all, data related issues, such as data transfer and network bandwidth are not taken into account. We will assume that the bandwidth required for data transfers is negligible and that it is unnecessary to use predicted bandwidth availability as a criteria for selection.

We also assume a homogeneous environment, where execution times for the same job will be equal regardless of the performance of the selected resource as our focus here is on reducing waiting times. In terms of policies, we assume that there is no pre-emption and priority is always in favour of the incoming GridSAM job. This implies that the maximum amount of time a job will have to wait is the amount of time that it will take for Dino or Sally to release a resource. It should also be noted that when a GridSAM service identifies an equal number of available resources, it will randomly select the target site.

The results obtained from our experiment are covered in Table 2. The GridSAM node that used Kalman filter prediction to determine where to submit to obtained on average a 135% improvement in job turn-around times over the meta-scheduler that relied solely on current utilisation meta-data, which is considerable.

| | Kalman Filter | Resource Availability |
|---|---|---|
| Avg. job length [sec] | 139.33 | 328.4 |
| No. jobs submitted | 100 | 100 |
| Submission interval [min] | 4 | 4 |
| Experiment duration [h] | 6.7 | 6.7 |
| No. jobs submitted to A (Dino) | 4 | 16 |
| Avg. length of jobs on A (Dino) | 620 | 1522 |
| No. jobs submitted to B (Sally) | 96 | 84 |
| Avg. length of jobs on B (Sally) | 125 | 108.9 |

**Table 2. Experiment results**

The main reason for the difference observed is the overall choice of resources. Jobs submitted to the cluster running Dino's workload (Cluster A) took considerably longer than those submitted to Cluster B due to the lengthy running times of Dino's jobs. However, because GridSAM node B had no knowledge of future resource availability, it failed to perceive that fact and submitted several jobs to that particular cluster at inappropriate times.

### 3.4 Control experiment

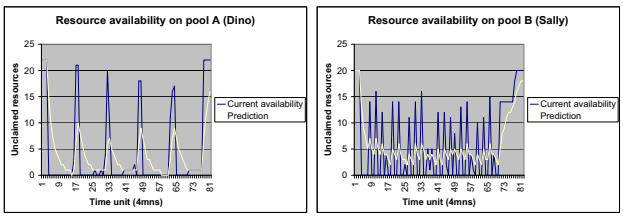


**Figure 7. Real vs. Predicted Availability**

As we have seen above, the use of Kalman filter prediction considerably improves the throughput of the meta-scheduler when there are clearly distinguishable patterns that can be forecast by the prediction service. However, as previously mentioned, we can expect that in scenarios where resources present similar patterns of use, little or no improvement will be made over the simple use of current availability, due to the minimal gain to be made in selecting one resource over another.

To illustrate this point, we detail here a control experiment, where we have adjusted Dino's workload in order to reduce the large gap in waiting times between each pool of resources. The result can be observed in Figure 7. Though there is still a difference between both patterns, this difference is no longer as marked as in our previous experiment. Dino's workloads are limited to jobs with a range of 15 to 20 minutes with larger pauses between re-submissions.

| | Kalman Filter | Resource Availability |
|---|---|---|
| Avg. job length [sec] | 167.53 | 226.3 |
| No. jobs submitted | 92 | 92 |
| Submission interval [min] | 4 | 4 |
| Experiment duration [h] | 6.1 | 6.1 |
| No. jobs submitted to A (Dino) | 39 | 52 |
| Avg. length of jobs on A (Dino) | 236.18 | 312.67 |
| No. jobs submitted to B (Sally) | 52 | 39 |
| Avg. length of jobs on B (Sally) | 119.2 | 116.76 |

**Table 3. Control Experiment Results**

As we can see from Table 3, we do still obtain an improvement of 35% when using Kalman prediction, though this improvement is not quite as significant as that obtained in the previous experiment. This is primarily due to the fact that waiting times on Cluster A (Dino) are not as long as our previous experiment – even though jobs submitted to that resource do take slightly longer. It is also interesting to note that when the GridSAM node with prediction capabilities did select that cluster, it did so less often than the node without prediction capabilities and at seemingly more appropriate times, since the average length of its jobs on that cluster is considerably lower. The fact that there is negligible overhead associated with the Kalman prediction – as the state maintained by the filters is minimal and the past history does not have to be maintained – implies that even when the improvement is only moderate, we can only benefit from the use of this technique.

## 4 Discussion and Related Work

Different approaches to meta-scheduling across organisational boundaries have been explored in the literature. Client side job-scheduling tools such as Condor-G [21], or the Community Scheduler Framework (CSF) [14], provide means of scheduling the submission of jobs to one or more grid resources – and remote job monitoring capabilities.

They do not or provide very basic resource selection mechanisms: Condor-G can for example perform basic matchmaking using user defined grid resource characteristics and a 'by the numbers' load-balancing technique that will ensure that only a specific number of submissions to a grid resource can occur simultaneously. Alternatively, the EzGrid broker [20] has the ability to take into account static and current dynamic parameters alongside policy information such as authentication and authorization policies. The EzGrid broker does not, however, attempt any form of prediction to determine the correctness of their observations.

Other approaches such as [5] have relied on economic incentives and models to provide brokering capabilities in a grid infrastructure. The focus, however, has been on maximizing economical efficiency in an environment where it is assumed that resources can guarantee a particular quality of service. We, on the other hand, assume a potentially variable quality of service due to multiple users sharing underlying resources, and make meta-scheduling decisions based on predictions of the level of service that will be obtained.

The specific use of predictions in grid and distributed environments has been explored by multiple researchers, though not necessarily for meta-scheduling purposes. In [8] the authors present a comparison between different classic linear models for time series forecasting used to predict load on a single Unix machine. However, the parameters of these models need to be selected by the user according to the characteristics of the particular time series of values that is taken into consideration. This process may be long and time-consuming since the user must first choose the model to use and then tune all the parameters for the specific time series. The use of state space models forecasting techniques does not require this setup phase; the model that we have envisaged can be applied to various deployment scenarios characterised by different utilisation patterns. Moreover, forecasting techniques are applied only to evaluate the load of a single machine and not in the context of cross-organisational scheduling in grid environments.

Similarly, auto-regressive methods, as explored in a grid setting by the Network Weather Services [23], also require a training phase to tune the parameters. These services can alternatively rely on mean and median based approaches to prediction, which would also require a training phase. The fact that historical data does not have to be maintained for our approach results in a lightweight method – whose applicability to global scheduling was demonstrated in an experimental manner.

Smith et al. in [19] use a classification method based on the similarity of sets of past workload traces. The authors classify each application using genetic algorithm searches to define good templates for workloads set of templates. Then they calculate the average running time for each type of template. The queue waiting times are predicted using these estimations. This technique is applied to improve the scheduling performance in a local environment. However, this approach cannot adapt dynamically to new patterns, since it is based on a finite set of templates derived from the previous workload history. It requires a possibly large set of observations for different types of applications to achieve valid estimations for each possible template from as statistical point of view. In an environment as dynamic as the grid, it is quite difficult to categorise the wide range of applications and users in advance, specifically when we do not have control over the environment.

An alternative predictive technique for scheduling of resources in parallel systems is presented in [24] based on the prediction of the future CPU loads. The approach is essentially based on the variance of the loads, assuming that the jobs have the same characteristics and this is not generally true in a Grid setting. In fact, according to our experience, as described in Section 3, different utilisation patterns can be observed. Moreover, a method purely based only on load variance can lead to very inaccurate estimations, especially in presence of high fluctuations [6].

## 5 Conclusions and Future Directions

We believe our findings have a number of implications for grid scheduling. Using an experimental approach based on existing workloads in a grid environment, we have shown that grid meta-schedulers and portals will be able to add real value to job submission by identifying the clusters that will allow users to obtain results more rapidly – encouraging the adoption of predictive scheduling and bringing the vision of the grid a step closer to reality. Furthermore, resource owners are now able to predict the time it will take before they are able to start executing a job relatively precisely, without any significant overhead. This paves the way for quality of service aware grid scheduling as resource owners are able to undertake quality of service guarantees without having to over-provision to the same extent as without such prediction. It also further facilitates the simultaneous use of resources across sites in settings where direct co-allocation through advance reservation is not directly supported.

By identifying the monitored resource parameters to satisfy a wide range of underlying resource management systems, and relying on potential combinations of parameters – for example by using a utility function – we can design flexible prediction mechanisms suited to heterogeneous grid and distributed environments a heterogeneous grid environment. We can also further refine our forecasting approach by adopting trend and seasonal components [3], which will most likely increase the accuracy of the prediction where periodic patterns are observed.

The use of predictions is an important building block for

a larger grid federation framework. In order to be able to aggregate resources effectively across organisations one must take into account the decentralised nature of the grid environment. The autonomous operation of individual resource management systems relied upon in a grid environment is pushing for the identification of novel approaches to compensate for the lack of direct control over resources and the potentially divergent behaviour of the underlying systems. As we have demonstrated, by relying on Kalman filter prediction techniques, we can considerably improve the use of resources in a global, and possibly autonomous, grid infrastructure.

## References


[1] P. H. Beckman. Building the TeraGrid. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 363(1833):1715–1728, Aug. 2005.

[2] F. Berman, G. Fox, and A. J. G. Hey, editors. *Grid computing: making the global infrastructure a reality*. John Wiley, 2003.

[3] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, 1996.

[4] R. P. Bruin, M. T. Dove, M. Calleja, and M. G. Tucker. Building and Managing the eMinerals Clusters: A Case Study in Grid-Enabled Cluster Operation. *IEEE Computing in Science and Engineering*, 7(6):30–37, 2005.

[5] R. Buyya, D. Abramson, and J. Giddy. NimrodG: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. In *Proc. of the* $4^{th}$ *International Conference on High Performance Computing in the Asia-Pacific Region*, page 283. IEEE Computer Society Press, May 2000.

[6] C. Chatfield. *The Analysis of Time Series An Introduction*. Chapman and Hall, 2004.

[7] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, FL*, volume 1459 of *Lecture Notes in Computer Science*, page 62. Springer, 1998.

[8] P. Dinda and D. O'Hallaron. An Evaluation of Linear Models for Host Load Prediction. In *HPDC '99: Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*, page 10, Washington, DC, USA, 1999. IEEE Computer Society.

[9] W. Emmerich, B. Butchart, L. Chen, B. Wassermann, and S. L. Price. Grid Service Orchestration using the Business Process Execution Language (BPEL). *Journal of Grid Computing*, 3(3-4):283–304, 2005.

[10] F. Gagliardi, B. Jones, F. Grey, M.-E. Begin, and M. Heikkurinen. Building an Infrastructure for Scientific Grid computing: Status and Goals of the EGEE project. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 363(1833):1729–1742, Aug. 2005.

[11] W. Gentzsch. Sun Grid Engine: Towards Creating a Compute Power Grid. In $1^{st}$ *Int. Symposium on Cluster Computing and the Grid*, page 35. IEEE Computer Society, 2001.

[12] C. Goonatilake, C. Chapman, W. Emmerich, M. Farrellee, T. Tannenbaum, M. Livny, M. Calleja, and M. Dove. Condor Birdbath - Web Service Interface to Condor. In *Proc. of the 2005 UK e-Science All Hands Meeting*. EPSRC, Sept. 2005.

[13] R. L. Henderson. Job Scheduling Under the Portable Batch System. In *Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 1995.

[14] Community Scheduler Framework (CSF). IBM contribution to the Globus toolkit. `http://sourceforge.net/projects/gcsf/`.

[15] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering*, 82(D):35–45, March 1960.

[16] W. Lee, S. McGough, S. Newhouse, and J. Darlington. A Standard Based Approach to Job Submission through Web Services. In S. Cox, editor, *Proc. of the UK e-Science All Hands Meeting, Nottingham*, pages 901–905. UK EPSRC, 2004. ISBN 1-904425-21-6.

[17] M. Musolesi, S. Hailes, and C. Mascolo. Prediction of Context Information using Kalman Filter Theory. UCL Research Note RN/04/22, University College London, Department of Computer Science, UCL, UK, 2004.

[18] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2002.

[19] W. Smith, V. Taylor, and I. Foster. Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 202–219. Springer Verlag, 1999.

[20] B. Sundaram and B. M. Chapman. XML-Based Policy Engine Framework for Usage Policy Management in Grids. In M. Parashar, editor, $3^{rd}$ *Int. Conference on Grid Computing, Baltimore, MD*, volume 2536 of *Lecture Notes in Computer Science*, pages 194–198. Springer, 2002.

[21] D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

[22] R. Tyer, M. Calleja, R. Bruin, C. Chapman, M. T. Dove, and R. J. Allan. Portal Framework for Computation within the eMinerals Project. In S. Cox, editor, *Proc of the 2004 UK E-Science All Hands Meeting, Nottingham*, pages 660–665. UK Engineering and Physical Science Research Council, 2004.

[23] R. Wolski. Experiences with Predicting Resource Performance On-line in Computational Grid Settings. *SIGMETRICS Perform. Eval. Rev.*, 30(4):41–49, 2003.

[24] L. Yang, J. Schopf, and I. Foster. Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 31, Washington, DC, USA, 2003. IEEE Computer Society.