Distributed Computing – Grid Computing

# Self-Organising Grid Resource Management

Statistical approach to delivering autonomous deadline scheduling

Key points

- Today's Grid workload contains predictable resource utilisation patterns
- Time-Series modelling can be used to forecast future resource needs and job execution times
- A novel meta-scheduler can use such predictions to support job scheduling to a user deadline

Maturity of activity: Prototype

Concept | **Prototype** | Trialed | Deployed

Concept | Prototype | Trialed | Deployed

Distributed Computing

# Deadline Scheduling on the Grid

## Objective

To enable execution of Grid jobs to a user requested deadline, without the need for extensive application changes or recompilation, hardware or software modelling, or resource profiling.
To minimise the administrative burden of Grid operators by offering an autonomous, self-managing deadline Grid scheduler.

## Characterisation of current Grid workload

The distributed computing and utility computing paradigms, with the Grid being one of their implementations, are increasingly being used to tackle a very wide range of jobs: from very demanding simulations of particle physics to a more mundane, repetitive tasks of serving network services or database lookups.

Successful management of Grid infrastructure on all time scales, from long term planning and provisioning to scheduling a single process, depends on good understanding of past Grid performance, historical load patterns and a decent anticipation of future requirements.

Our approach looks at meta-data associated with each Grid job, such as submitting user, time of day, executable name and similar, trying to identify patterns of similar behaviour. Once a certain “class” of jobs is identified (perhaps a scientist submitting one type of simulation from a certain workstation throughout a working day) further submissions with similar meta-data can be associated with historical behaviour of that respective job class.

## Autonomous deadline Grid scheduling

To support a deadline scheduling method which is largely autonomous and self-managing, we have opted for a statistical and probabilistic approach to estimating future job running time and resource requirements. Each “class” of jobs is modelled using time-series methods which are used to give forecasts and associated confidence levels.

In a large, multipurpose Grid facility, providing utility-style computing for a large number of users with varying profiles of resource requirements, this approach can produce a probabilistic guarantee of deadline adherence, without assurance of a particular job finishing on time.

## Progress

We have analysed workloads from different academic Grids and have seen significant levels of correlation of meta-data and execution times in all of them. Auto-regressive and moving-average methods have produced good forecasting models, especially in conjunction with our anomaly detection algorithms. We see this work complementing current Grid schedulers and ongoing research on Grid economy models.

Contact:
Aleksandar Lazarevic: a.lazarevic@ee.ucl.ac.uk

# Deadline Scheduling on the Grid


Aleksandar Lazarević
PhD Student
E: a.lazarevic@ee.ucl.ac.uk W: www.ee.ucl.ac.uk/~alazarev/
Supervisor: Dr. Lionel Sacks


## The Grid

The Grid is a generally accepted term for wide-area distributed and heterogeneous fabric of computational, storage and scientific resources. Depending on the field of use, Grids are implemented to deliver high-throughput / high-performance computing, hardware virtualisation or increased resilience. In the context of my research, the Grid is a seen as a future computational platform delivering different next-generation network services to a wide range of users with varying resource requirements and utilisation profiles.

## Deadline Scheduling

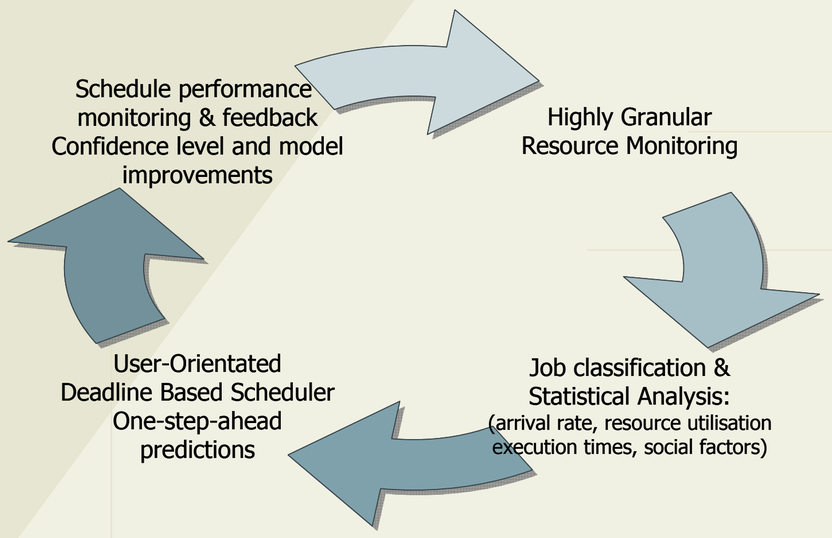
For reasons of backward compatibility and robustness, scheduling on the Grid has so far been a direct transplant from legacy batch systems. Despite often being the easiest solution, these schedulers are a poor for a utility platform, with highly dynamic resource pool and a wide variety of users with different usage profiles.

Our approach was to offer users an opportunity to have their jobs finished to a deadline and a notional cost. This forms a foundation for Grid economies in which resources are traded to deal with peaks and troughs of demands, and can more naturally translate user-operator relationship (such as Service Level Agreements) from the business to the technology domain.

## Delivering Deadlines

A large base of scheduling research has produce some highly effective space-filling algorithms ensuring scheduling optimisation to increase hardware utilisation, reduce makespan or shorten queue wait times. But all these algorithms depend on a prediction of job resource requirements and execution time. These have so far been delivered using methods requiring source code re-compilation, specific hardware profiling or application instrumentation. All of them are either too computationally expensive, of low quality or of limited in scope.

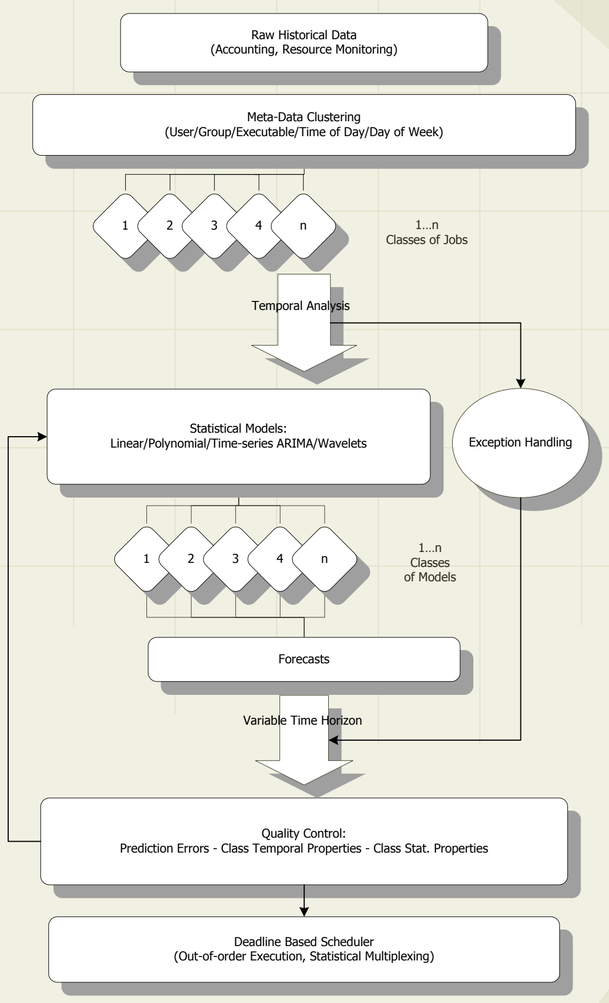
Our work aims to deliver probability density forecasts by applying statistical models to the historical job accounting and monitoring data. The approach is largely autonomous and self-managing and requires no changes to the application, underlying hardware or administration process.




## Architecture

Our logical architecture (**left**) is based on autonomous development of statistical models based on the systems current state and users' historical behaviour. Forecasting quality is constantly monitored and used to adjust confidence levels and as a feedback into model optimisation.

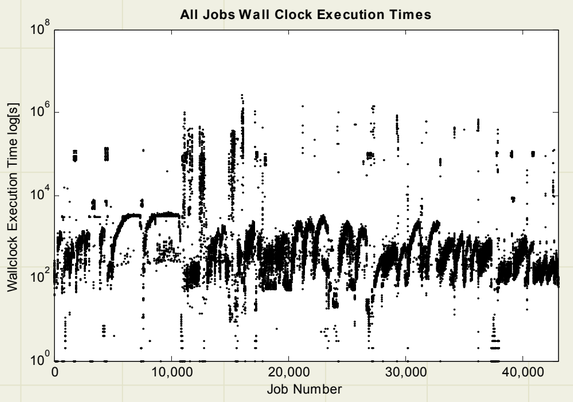
Implementation architecture (**right**) is based around three main stages:
1. **Meta-data Clustering** analyses additional information about the job (such as username, group, executable name) and groups the jobs according to similar utilisation patterns.
2. **Statistical Modelling** applies a number of different time-series models to the grouped data, depending on its complexity and predictability.
3. **Exception Handling** deals with sudden and unpredicted changes to the job profile that give rise to significant and sustained forecasting errors.

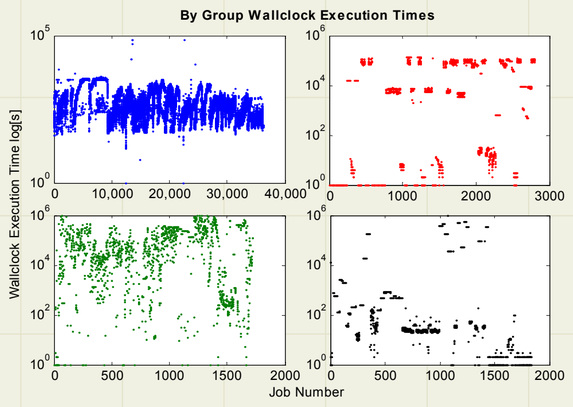


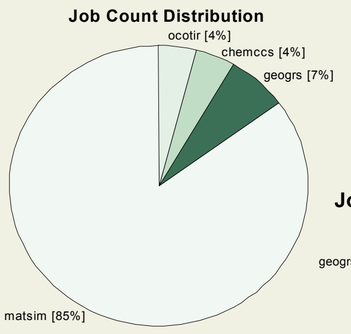

## Workload Characterisation

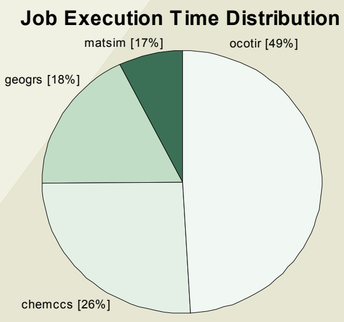
Understanding of statistical properties and characteristics of today's Grid workload is essential in developing a probabilistic scheduler. We have analysed 2 years of data from UCL's Grid facility, and are now running similar tests on publicly available job traces from other research Grids.

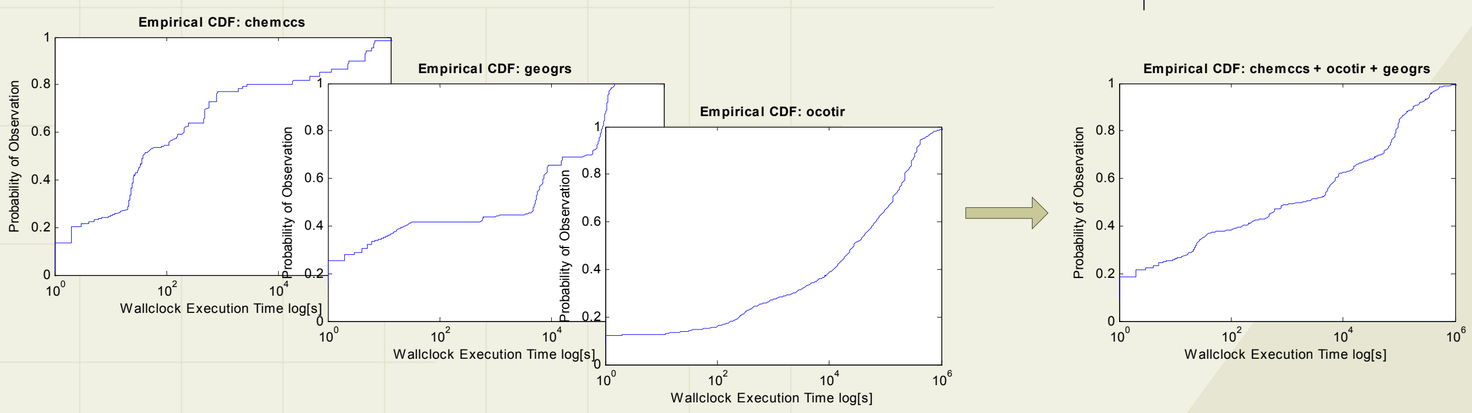
The meta-data clustering (**left)** based on the job submitter's credentials shows a significant difference in execution times and their distribution.
A contrast between the number of jobs submitted and their execution time (**right**) means a complex model may not be suitable for a job group with a high number of jobs contributing to a small proportion of overall resource utilisation.
Cumulative Distribution Function plots (**below**) show the probability of observing jobs of duration equal to or less than indicated on *x* axis. Despite markedly different profiles of the three user groups plotted, their aggregate effect is a good fit to a straight line: every execution time is as likely to be seen as any other.




## Predictability Testing

Overall quality of forecasts will depend both on the prediction algorithm as well as the underlying predictability of the job data. Execution times are multi-modal and contain significant epoch changes at which point the distribution's properties change beyond the model's tracking ability. In between these changes a model can be developed to track execution times with significant accuracy.

It is important to note that spot prediction accuracy is secondary to the ability of the model to estimate the confidence bounds and offer forecasts with as little variance and systematic over- or under-estimation as possible. In an environment with a large number of uncorrelated jobs, and probabilistic quality guarantees, spot prediction errors will cancel each other out.

Two predictability test plots are show: AutoRegresive-MovingAverage (**left**) and 3rd order polynomial (**right**). The data is an actual trace from UCL's Grid, and errors are based on the algorithms estimated execution time and real execution time from the trace. In this particular case, ARMA predictor is superior but much more complex – by monitoring the scheduling performance our scheduler would be able to trade-off quality and performance as required by higher level policies (SLAs).