

# GridSHED

## e-Science Solutions for: Grid Scheduling, Hosting and Environment Design

This project investigates some key issues involved in developing commercially viable Grid hosting environments, focussing on system modelling and middleware development:

### 1. System Modelling

Detailed stochastic modelling is used to determine strategies for maximising efficiency. Three key areas will be modelled: Dynamic resource allocation, Routing, Resource discovery and dynamic configuration.

#### Dynamic Resource Allocation – A Simple Model

Our model consists of two types of jobs, which are submitted to a pool of  $N$  servers. Configuring these servers involves dedicating  $k$  of the servers to type 1 and  $N-k$  to type 2, as illustrated in Figure 1. Configuration decisions are traditionally based on statistical analysis of the arrival rates and expected lengths of jobs. This results in a static configuration, which cannot react to changing demand. Each server can be reconfigured to accept jobs of other types, but such switches are costly. Long delays for type 1 or type 2 jobs are also costly, so a dynamic policy is required to determine the most effective server configuration, based on current demand. Switching decisions are *memoryless*, depending on the current system state but not on past history.

Dynamic programming techniques have been used to investigate switching decisions and produce look-up tables of optimal policies. However, these

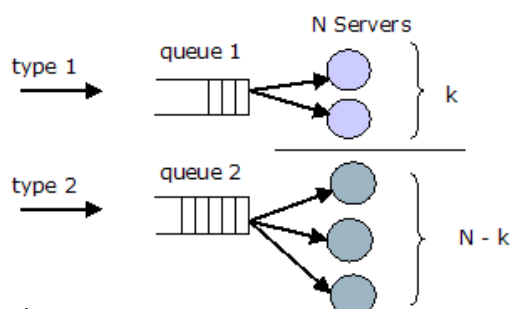


Figure 1

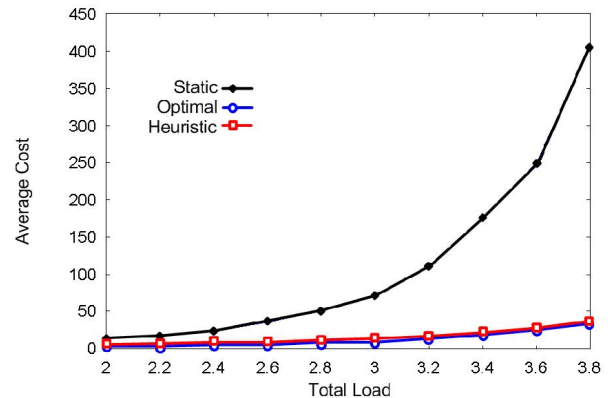


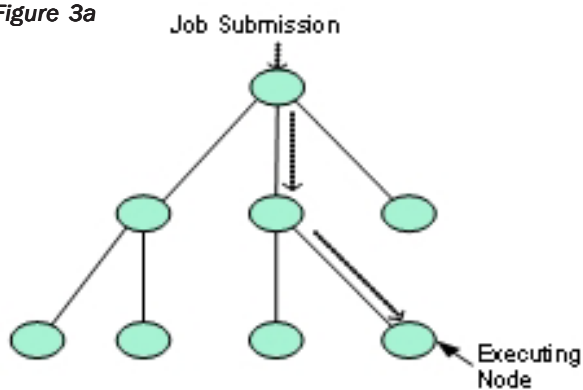
Figure 2

are computationally expensive to produce and store, being highly dependent on parameters such as the arrival rate of demands and average job length. Ideally, the optimal policy would be characterised explicitly in terms of the parameters but this is not always possible for such complex problems. Instead, we have formulated an heuristic policy that is simply characterized and performs acceptably well compared with the optimal policy. The graph in Figure 2 shows the advantage of dynamic reconfiguration – storage costs have been dramatically reduced as overall load of the system increases. We see that our heuristic performs extremely well compared to the optimal policy. Future work within dynamic resource allocation will include the natural generalisation of the problem to consider more complex scenarios.

#### Routing, Resource Discovery and Dynamic Configuration

The two remaining key modelling areas are yet to be addressed. In larger systems, a number of scheduling/resource management problems might occur, including routing job requests to appropriate pools for maximum efficiency and designing system architectures to most effectively reduce storage costs and job waiting time. Detailed stochastic modelling will be used to help maximise the efficiency of hosting environment design and dynamic changes to the architecture will be exploited for further efficiency gain as demand varies.

Figure 3a



## 2. Middleware development

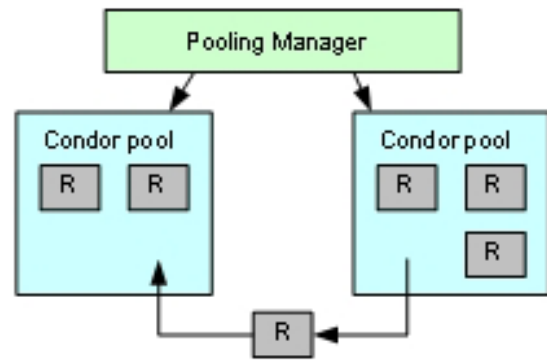
The project will develop Grid services providing efficient management of resource pools and demonstrating the legitimacy of the mathematical model. Resources may be switched between pools to help the system cope more effectively with unpredictable demand.

### Architecture

A large amount of data must be obtained, stored and continually processed to ensure effective resource reallocation. As a hosting environment may contain thousands of machines, it would be inappropriate for a single entity to manage the whole system due to problems associated with bottlenecks and fault tolerance. A solution to the problems of a centralized approach is to employ a tree or network structure, with each node containing various pools, and with resources belonging to those pools switching only within a node. This limits the potential for resource switching, but eliminates core problems of a centralized approach and simplifies management of heterogeneous hosting environments.

Jobs may be submitted at any node on the tree. A node may either process the job itself, or query its children, which will each return a cost value based on their current state and the nature of the job. The parent may then submit the job to that node offering the lowest cost. This may be recursive, so children receiving a cost query may in turn query their own children. Tree nodes contain various pools of resources. A pool manager at each node provides the following services: calculating the cost of running a job on its various pools; determining to which pool a submitted job will be sent; and deciding whether resources should be

Figure 3b



switched between pools. Figure 3a shows a job entering at the top of a tree and being directed along the most appropriate nodes, to be processed at the node with the least cost.

### Initial Prototype

The prototype focuses on switching single resources (machines) between pools. Each pool in the system processes different types of jobs, job types being based on factors such as operating system requirements. This mirrors real-world hosting requirements, where servers may switch OS on-demand. Resources are switched or not based on various metrics gathered from the system, including queue lengths, average job arrival rates, execution times for job types, the number of resources available in each node, and the time taken to switch between job types. Each pool is managed with the Condor resource manager, which makes routing decisions and maintains job queues. During prototype development, Condor's job description language was extended to include attributes particular to the system (such as a job ID and job type). Condor was chosen for simplicity and expediency of prototype development; future versions will operate within a heterogeneous hosting and resource allocation environment. Figure 3b depicts the resource allocation structure within a tree node. The pooling manager continuously monitors the pools in the node, switching resources between pools when necessary. The next stage of development will involve redeploying the prototype within a Grid Services framework and extending it to function within a heterogeneous resource management environment. The services will then be integrated with technology at the industrial partner's site.