Grid Resource Allocation: Allocation Mechanisms and Utilisation Patterns

Stefan Krawczyk and Kris Bubendorfer

School of Mathematics, Statistics and Computer Science
Victoria University of Wellington
Kelburn, Wellington, New Zealand
Email: kris@mcs.vuw.ac.nz

Abstract

Grid systems have been put to remarkable use in recent years. Finding planets, rendering multi-million dollar movies, and helping to understand disease are just some of the examples grid systems have been used for. With business turning to towards using grid systems and looking to make them global mechanisms for service delivery, they are nicely poised to be an exciting future prospect. However the performance of a grid system is strongly related to how well grid resource allocation is performed. With many possible approaches to grid resource allocation we have to ask the question, what impact does the choice of resource allocation methodology have on the utilisation and performance of a Grid system? This paper addresses this question through the investigation of the characteristic allocation patterns for three different resource allocation mechanisms and their subsequent effect on resource utilisation within a simulated Grid system.

1 Introduction

Grid computing has not become an important field of research in computer science, but also contributes to the advancement of science in many other fields by providing access to large scale shared computing resources on which to solve computationally large problems. The Grid is a distributed computing infrastructure(Foster et al. 2001) that takes advantage of many networked computers to compute large scale problems. It has been used in high throughput computing e.g. High Energy Physics (HEP) applications, large-scale resource and data sharing (Earth System Grid), and on demand computing (Sun Microsystems 2007). It has grown from having a focus on advanced science and engineering to one that includes the potential for commercialisation by both industry and individuals (Dimitrakos et al. 2003).

This paper seeks to answer the question *"what impact does the choice of resource allocation mechanism have on the target Grid system?"* by examining how different allocation mechanisms change the utilisation of a simulated Grid system. We have selected three different allocation mechanisms that typify current thinking in resource allocation and have constructed a GridSim (Buyya & Murshed 2002) simulation on which to observe their performance. We do not consider local scheduling, our focus is on allocation mechanisms that operate above GRAM (Feller et al. 2007) (or equivalent) level. We do not consider workflow scheduling (Thain et al. 2005) or advanced reservation (Netto et al. 2007), as these would serve to hide the very characteristics that we interested in observing by placing application or user constraints on the allocation mechanism. We also do not look at scheduling systems that delegate fine-grained scheduling within larger scheduled blocks such as those allocated by Falkon (Raicu et al. 2007).

This paper is organised as follows: we firstly give an overview of allocation in Grid systems, we then look at the protocols that we have selected to investigate and the design of the simulation, we then present the characteristic utilisation patterns that we observed and what observations and conclusions that we can draw from this data.

2 Background and Resource Allocation

Before talking about the different models and mechanisms for allocating resources in the Grid, we must first define what the Grid actually is. Until the paper *The Anatomy of the Grid - Enabling Scalable Virtual Organisations*(Foster et al. 2001) there was significant doubt as to what 'grid computing' or 'grid systems' actually referred to. From this paper we can take the foremost principle that what defines a Grid system is that it facilitates resource-sharing among a set of participants (some provide resources, others consume them). The shared resources are then put to use by some of the participants. The resource providers and consumers have clearly defined conditions in which they interact e.g. what is shared, who is allowed to share and what quality of service one might expect. The term virtual organisation (VO) is used to describe participants who interact under these conditions.

In essence a Grid is expected to encompasses three points (Foster 2002):

- The resources co-ordinated are not subject to centralised control, i.e. they run under the domain of a virtual organisation.
- The standards, protocols & interfaces used are standardised, open and general purpose, i.e. the interoperability of the resources allows seamless integration with anything.
- The quality of service delivered is not trivial i.e. to an agreement, to different types of agreements, etc.

The remaining parts of this section will now look at three different styles of Grid resource allocation. We will look at BOINC (Anderson 2004), FairShare (J.Kay & P.Lauder 1988) an agreement based allocation and an economic allocation scheme based around a reverse first price sealed auction. The details of the actual algorithms will be provided in later sections 2.6.1.

2.1 Volunteer Resource Allocation

Volunteer computing or public resource computing is based on the idea that willing participants with idle resources, like CPU, will happily donate these in the aid of some cause or task without any tangible remuneration for their use. It is characterised by the ease of entry and exit of resource providing participants in sharing their resources. Typically resources are allocated to volunteers on a pull basis, with blocks of tasks being allocated on request, and collected when the next set of tasks is requested.

To see the effectiveness of volunteer computing in providing tremendous amounts of computing power, one has to look no further than the SETI@home(Anderson et al. 2002) project. In 2001, the average computing throughput provided by volunteers was 23.76 Teraflops! There are a many other BOINC (Anderson 2004) based projects including Rosetta@home, TANPAKU@home, LHC@home, etc. One social change that is observable in the volunteer computing environment is that a high computing to data ratio is becoming less important, and therefore the advent of broadband has the potential for data intensive applications to benefit from this model. However, whilst some might argue that BOINC does not strictly meet the criteria established for defining a Grid due to the topology (star or similar (Sarmenta 2001)), the work is often published at Grid conferences and provides a usefully different allocation mechanism (Anderson et al. 2005) for the purposes of this paper.

2.2 Agreement Based Resource Allocation

Agreement based resource allocation is a model where the negotiation and management of resources is formed by set policy. In essence either through negotiation (automated or not) a set of policies setting the requirements or regulations for resource are established and used for the allocation decisions. Each participant adheres to the policy set out. This form of resource allocation is usually used in an organisation on its own grid, between collaborating institutions, etc, i.e. in general between places where collaboration is high and long-term, e.g. universities and closed communities.

Agreement Based resource allocation comes in two general forms:

- Policy based resource allocation is a model which uses a hierarchical approach to recursively subdivide the resources available. For some, logically dividing the grid resources makes sense, as this way everybody has access to them. This is heavily linked to the concept of virtual organizations (VO) (Foster et al. 2001). Using a policy, a share of the resources is allocated to each VO. The VO can then be made to enforce a lower level policy or just subdivide its resources among its members. Variants of policy based resource allocation include Fairshare scheduling (J.Kay & P.Lauder 1988) and decentralised Grid-wide Fairshare scheduling (Elmroth & Gardfjall 2005). These two are focused on grid resource utilisation and aim to deliver the allocated target shares through policies.

- Service level based resource allocation (Andreozzi et al. 2005), as the name suggests, relies on the use of service level agreements (SLAs). Each grid user must produce a service level agreement indicating their expectations of service quality. A resource broker then performs some match making to assign suitable resources to requests. With the service level agreement, a better quality of service is enforced. This is an active area of definition being defined by the Grid Resource Allocation and Agreement Protocol Working Group of the Global Grid Forum (*Grid Resource Allocation and Agreement Protocol Working Group of the Global Grid Forum* 2003).

2.3 Economic Resource Allocation

Economic resource allocation is where the negotiation of resources is done via an economic mechanism. A currency is used as the medium of exchange, independent of whichever economic mechanism is in place. As is seen in modern economies, this greatly enables the exchange (or at least a chance to) of needs and wants in return for goods and services. Auctions have long been suggested as a means to allocate resources in a distributed system (Malone et al. 1988) whereby clients initiate an auction to find the best offer of resources to execute a task. Indeed the earliest example of a computational resource auction that we have found is (Sutherland 1968) in which a whiteboard based auction was utilised by which bidders could spend their processing time allocations.

The main advantage of auctions for distributed and Grid computing is that they are naturally decentralised which permits many auctioneers to function in a distributed network. Auctions can also compute optimal allocations giving providers the best return on resources. There are four main types of auction protocol; the English, Dutch, Sealed-Bid, and the Vickrey auction protocol. The English auction is the conventional open outcry, ascending price, multiple bid protocol. The Dutch auction is an open outcry, descending price, single bid protocol. The Sealed-Bid, or tender, is a sealed single bid, best price protocol. The Vickrey auction is similar to the Sealed-Bid auction, except that the winning bidder pays the amount of the second best bid. All four auction protocols yield the same return in private value auctions hence the selection of an auction protocol usually depends on messaging and other implementation requirements (Vickrey 1961).

Economic resource allocation is seen as the mechanism that will enable real growth in the number of Grid systems, in particular commercial ones. This will lead to more choice and better options for potential Grid consumers. As Buyya, Abramson, and Giddy state *"It offers incentive for resource owners to be part of the Grid and encourages consumers to optimally utilize resources and balance time frame and access costs."* (Buyya et al. 2001)

Another reason for its attractiveness is that it seems the most intuitive. As modern day economies revolve heavily around trade and allocation of resources, which is done via economic mechanisms, implementing this mechanism would seem the most natural. Experience with these mechanisms also gives us knowledge with which we can theorise what the effects in grid systems would be (R. Wolski and J. Brevik and J. S. Plank and T. Bryan 2003). The mechanisms proposed for use in grid resource allocation include auctions, commodity markets, tenders and posted price (Buyya 2002).

With the emergence of Ebay and other Internet auction sites, it is easy to see that particular economic mechanisms are better suited than others to selling or getting (depending on your perspective) goods and services in a particular context. For instance market gardeners use a commodity market mechanism to sell their produce, but at the supermarket we use a posted price mechanism.

Therefore what economic resource allocation mechanism should be used in grid systems? Different mechanisms result in potentially different market properties, like price stability, grid resource utilisation, efficiency, etc. This means that choosing the right mechanism(s) would be of utmost importance for the success of the grid system. There are also issues with a closed or open currency (Parkes et al. 2001) and trust (Bubendorfer & Thomson 2006) that are beyond the scope of this paper. Various types of auctions and commodity market models have been simulated including (R. Wolski and J. Brevik and J. S. Plank and T. Bryan 2003, Kant & Grosu 2005, Das & Grosu 2005, Regev & Nisan 1998, Pourebrahimi et al. 2006).

2.4 The Simulation

We cannot at present answer the general question: *which of the above allocation mechanisms will return the greatest utilisation of the Grid system?* This is not only because we lack any substantial evidence (empirical, proof, or simulated), but also because the approaches differ in their ability to control the resources. What we can do however, is observe the utilisation patterns, and therefore make more informed choices when designing a Grid allocation system to best match the usage requirements.

The first obvious step is to build a simulation which will help determine at least how the choice of mechanism will impact or characterise the utilisation that a Grid system will deliver. The advantages of simulation are well understood, and provide a flexible and controlled environment in which to carry out comparative analysis. This greater control allows better analysis of underlying behaviours and their causes.

2.5 Simulation Package

Simulation is a tool for system analysis that has been studied for a considerable length of time. In this respect it is a well understood discipline, with well defined areas of application. Shannon (Shannon 1975) provides the following formal definition: *Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system.* The important point is that this is without the costs inherent in building or modifying a real system. Simulation can also deal with more complex systems and interactions between systems than analytical models.

At the time of writing this paper, there were three published simulation packages that enable simulations of some type of resource allocation for the grid: GridSim(Buyya & Murshed 2002), SimGrid(Casanova 2001) and Mercatus(Kant 2005). Both GridSim and Mercatus are implemented in Java and run on top of SimJava(Howell & McNab 1998), which is a process based discrete event simulation package. SimGrid on the other hand runs on its own components and is implemented in C. The GridSim simulation package was chosen as the environment to perform the simulations as the most developed overall package.

2.6 Simulation Design

There were three types of resource allocation mechanisms simulated: an implementation of volunteer resource allocation, which will be called Volunteer Pooling, FairShare resource allocation which is Policy based resource allocation, and a reverse first priced sealed auction (RFPSA), which is economic based resource allocation.

The general design of each mechanism follows the diagram described in Figure 1. This is the most commonly seen form in literature(Buyya et al. 2000, Buyya 2002) for Grid Systems. Greater detail about the simulation design of the mechanisms can be found in (Krawczyk 2006).

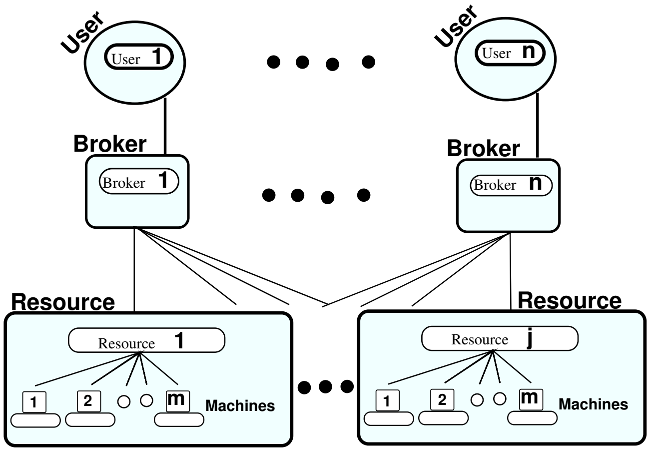

Figure 1: General Structure of the Implementation. Each solid line between an entity represents a flow of jobs.

2.6.1 Volunteer - Volunteer Pooling

The volunteer computing allocation algorithm used in our simulation differs from standard in two significant ways. Firstly volunteers are pooled together, which form several grid resources to better fit the general Grid approach. Secondly reliability is delegated to each resource pool, which is responsible for making sure that a job that is assigned to the resource pool is completed, even if an individual machine fails. The term failure refers to a machine that has gone offline/been disconnected/chosen not to continue processing the job. In standard BOINC (Anderson 2004) redundant computations are used to resolve these forms of failure. In our simulation we have made the assumption that no volunteer machine is malicious, so once jobs are done they are not verified.

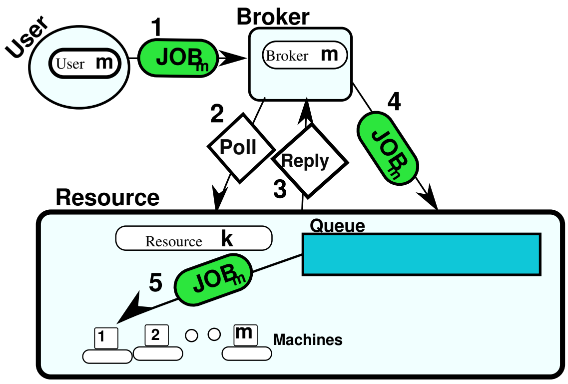

Figure 2: Volunteer Pooling mechanism steps.

The various roles in the allocation algorithm are (refer to Figure 2):

- **Users** produce the jobs that are to be executed by the grid system. They each have a broker which they send these jobs to at a constant rate (step 1), until they have none left.

- **Brokers** on receiving a set of user jobs, polls all the available resources and waits for a response as to how many jobs the resource can take (steps 2 & 3). The broker distributes the jobs around to random responsive resources until it has none left (step 4).

- **Resources** on receiving a subset of jobs, assigns jobs to any free machines (step 5), and then places any remaining jobs in a queue. Jobs are serviced on a first in first out basis from the queue. Should a volunteer machine crash and fail to do a job, the resource either reassigns that job, or puts it back in first position in the queue.

2.6.2 Agreement - Policy based - FairShare

Out of the agreement based resource allocation mechanisms, the decision was made to implement a policy based one, namely a fairShare variant based on the Grid-wide Fairshare(Elmroth & Gardfjall 2005). The implementation is simplified in that it only has one policy level, i.e. there is only a top level representation of users and no distinction, such as sub-users, for which to have a second level of policies.

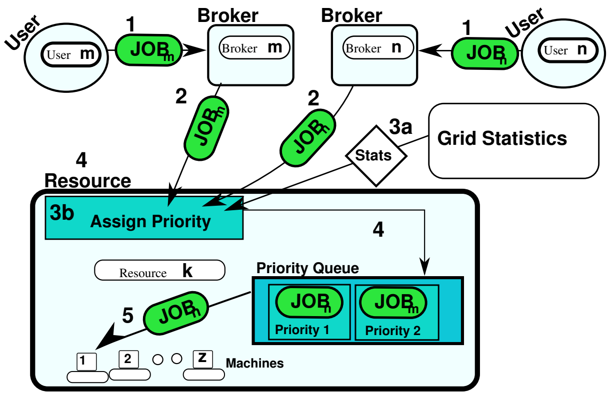

Figure 3: FairShare mechanism steps.

The various roles in the allocation algorithm are (refer to Figure 3):

- **Users** Each user in FairShare is assigned a target share of the grid resources. This is set at the beginning of each simulation and does not change. Again users produce the jobs that are to be executed in the grid system. They each have a broker which they send the jobs to at a constant rate, until they have none left (step 1).

- **Brokers** Once a broker receives a set of jobs, a resource is randomly picked and assigned a subset of the jobs (step 2). The broker is limited as to how many total jobs can be outstanding at any point in time. This stops the resources from being flooded with jobs.

- **Resources** Once at the resource, the job is assigned a priority based on the deviation from its user target share and the current share supplied by the GridStatistics entity (steps 3a & 3b). It is then put into a priority queue (step 4), and the first job at the queue is the next job to be processed (step 5). Each time the users current shares are updated, all the jobs in the queue have their priorities updated as well. In the simulation the current user share used for the priorities takes the total usage history of the grid system into account rather than decaying as in the real protocol.

2.6.3 Auction

The reverse first price sealed auction (RFPSA) was implemented, as it is a simple, quick auction. Jobs are put up for tender, and the lowest bid wins. In total seven different auctions were simulated. The differences between them were based on their reserve price setting policy and their job costing policy. For the sake of clarity, space and interest, only three of the auctions will be presented. This is because these auctions were the most interesting, and the others were the same if not similar. The basic auction was implemented as standard with the GridSim distribution - although we did create the auction costing and bidding policies.

In short, the different auctions were (refer to Figure 4 for mechanism steps) :

- **Auction df_load-dif** had a broker which was drip fed currency, and the resources responded by changing their bidding price based on the load difference they experienced.

- **Auction Random** had a broker that had no budget, and its resources bid a random number.

- **Auction Load** had a broker that had no budget, and its resources returned the current load it was under.

- **Auction WaitingT** had a broker that had no budget, and its resources returned the current average waiting time of jobs currently processing.

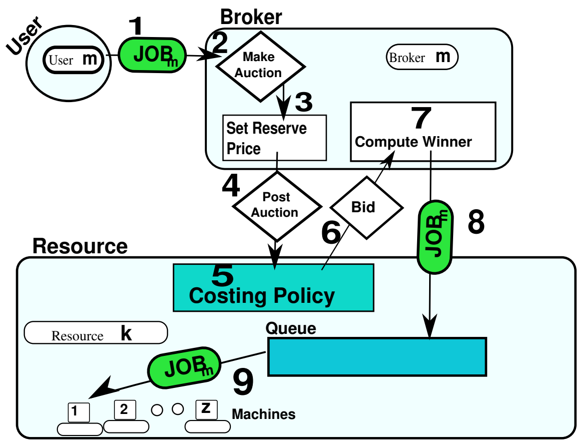

Figure 4: RFPSA mechanism steps.

- **Users** produce the jobs that are to be executed by the grid system. They each have a broker, which they send these jobs to at a constant rate, until they have none left (step 1)

- **Brokers** On receiving a job, the broker proceeds to make a RFPSA for it (step 2). The reserve price that it sets is dependent on the price setting policy chosen (step 3). In policy drip-feed (df), the broker had no budget to start with, but was given budget injections at the same rate at which jobs were coming in. The reserve price was set at:

$$\frac{money\ from\ injections - money\ spent}{jobs\ waiting\ at\ broker + jobs\ being\ auctioned}$$

Auction df load-dif used this policy. This was to recreate a broker being drip fed currency (e.g. tokens or money), and using the available currency (what it had not managed to spend) split over the jobs that it currently had. In policy no-budget, the broker had no budget. The reserve price was not set, indicating that the broker would accept anything. Once the reserve price was set, the auction was posted (step 4). After the auction was over, the broker would computer the winner (step 7) and then send the job (step 8) to the winning resource.

- **Resources** Resources first receive notification that an auction is taking place. The method that the resource uses to cost the job is dependent on the policy chosen (step 5). There is not space to reiterate all the policies, so only the least intuitive will be presented here, the remaining policies can be found in (Krawczyk 2006). In policy load-difference (load-dif) the price bid by a resource increased or decreased depending on the difference in the load the resource was under from the last time it checked. Specifically load difference was calculated as:

$$\begin{aligned} &lastload \\ &- (\Sigma(processing\ time\ of\ jobs\ in\ the\ queue) \\ &+ \Sigma(processing\ time\ left\ of\ executing\ jobs)) \end{aligned}$$

  The resource would then produce a bid for the auction (step 6) and if it happened to win, would allocate the job on a FIFO policy (step 9).

2.6.4 Simulation Metrics

For the simulations several metrics were implemented to gauge a mechanisms performance. These were:

- Recording of statistics on the total grid utilisation (total used computation time/total available computation time).
- Recording of statistics on the amount of time a job spends at a broker.
- Recording of statistics on the amount of time a jobs waits at a resource before being processed.
- Recording the finishing time of each simulation.
- Calculating averages, medians, standard deviations and inter quartile ranges where possible.

2.6.5 Simulation Parameters

The number of users, number of resources, number of jobs, size of jobs, amount of machines, amount of processors a machine had and their processing power, the budget for brokers in auctions, and the rate at which jobs are sent from the user, are all potential variables. The decision was made to:

- Set the amount of processing elements at one per machine.
- Keep the total amount of machines and their processing power fixed at 100 machines, and 400 MIPS.
- Base the job length on a Poisson distribution with a mean of 400 × 100 machine instructions, which would take 100 seconds.
- Keep the total amount of jobs the same.
- Base the job rate on a negative exponential distribution with mean equal to two seconds.
- Keep the unit price of a job for auctions the same at twenty five.
- Vary the number of users, while keeping the number of resources fixed.
- Vary the number of resources, while keeping the number of users fixed.

2.7 Experimental Configuration

Varying the number users shows the sensitivity of the scheduler relative to the user load on the Grid system, while varying resources shows the sensitivity of the scheduler relative to resource availability. A more complete set of results is available in (Krawczyk 2006).

- **Number of Users** In the first set of experiments, the number of users was varied between one, two, five, ten and twenty users. The number of resources was held fixed at five resources and twenty machines.
- **Number of Resources** This set of experiments looked at varying the number of resources. The number of resources was changed between one, two, five, ten and twenty resources. The total amount of machines in the grid system was always evenly distributed among the resources and totaled one hundred. The number of users was fixed at five users based on the previous results.

2.8 Results and Analysis

In this section we will start by examining the utilisation characteristics of the scheduling algorithms. Figure 5 shows the utilisation achieved using Volunteer Pooling, with no failures. In this case the best performance is achieved with one user, and as the number of users increase, the completion times correspondingly increase. Scheduling is completed by 3250 gridsim seconds, yet the final job (for 20 users) does not complete until 15,000 seconds. These results are directly attributable to the uneven distribution of jobs. In addition, the performance is considerably worse when we add failure of jobs, see (Krawczyk 2006).

Utilisation in FairShare is considerably better as shown in Figure 6. As the number of users increases, the utilisation experiences a sharper initial increase in utilisation. Utilisation remains high until all jobs complete – the algorithm is essentially independent of the number of users. For the single user, the job submission limit (number of outstanding jobs permitted) artificially limits the utilisation – however, this was constant for all of the algorithms tested.

The final utilisation graph in Figure 7 shows the characteristics of the df_load-diff auction. You can see that the initial knee of the utilisation graph is softer than for FairShare, and indeed is also softer than that achieved by Volunteer Pooling. This is attributable to the inherent bidding delay (closing time) experienced by auctions. This small amount of time wasted adds up and results in the auction's *soft start.* The

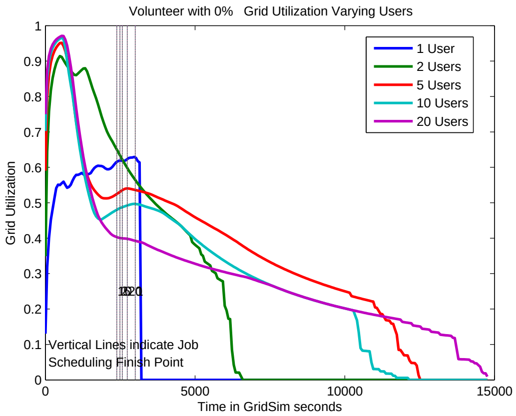

Figure 5: This graph shows the poor utilisation achieved by Volunteer Pooling for 1,2,5,10 and 20 users.

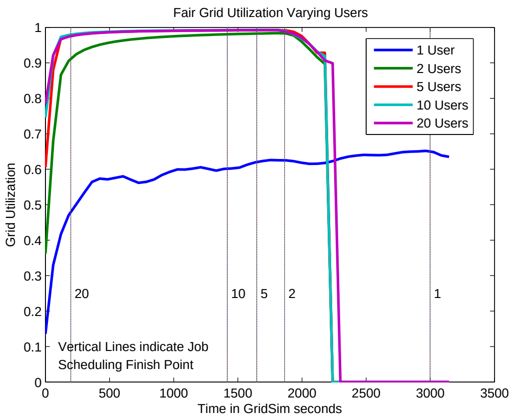

Figure 6: This graph shows the high utilisation achieved by FairShare for 1,2,5,10 and 20 users.

impact of the *soft start* is lost once jobs start to be put into queues at the resources. This is probably the most interesting utilisation characteristic revealed, as it holds for the entire class of auction mechanisms. Therefore an auction should preferably be deployed in a continuous scenario. The submission limit for a single user has the same impact as in the FairShare algorithm.

The next set of results we look at the processing time versus the waiting time. The more consistent the algorithms, the cleaner the more regular the scatter plots are. The first graph in Figure 8 shows the wildly inconsistent waiting times experienced with the Volunteer scheduler. The horizontal bands in this graph reveal that the resource queues were not even, and show that the distribution of load is very uneven – corroborating the utilisation results in Figure 5.

The graph in Figure 9 shows the processing time versus the waiting time for the FairShare scheduler. There are three identifiable clusters corresponding to the number of users. The horizontal component (waiting time) is highly variable although less so than the result for the Volunteer scheduler given in Figure 8. FairShare's lack of waiting time consistency (compared to the best auctions) is probably due to its policy specifying that each user receives an equal share of the grid resources. Had its policy specified to focus on the maximum job waiting time, then its results probably would have experienced the job waiting time stability of the auctions.

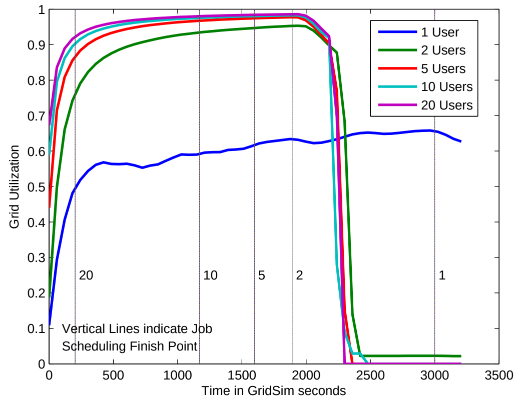

Figure 7: This graph shows the utilisation achieved by the df_load-diff auction algorithm for 1,2,5,10 and 20 users. Note the softer knee - characterising the soft start of all the auction algorithms studied.

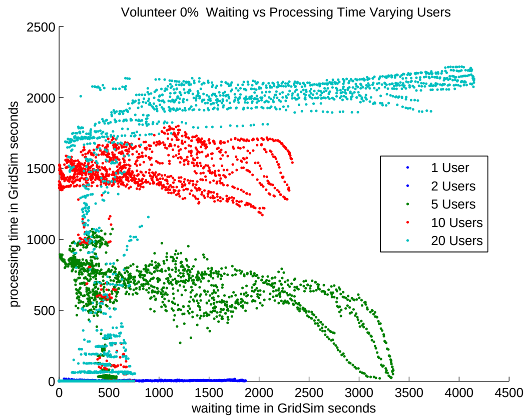

Figure 8: This graph shows the poor distribution of load to resources and the subsequent inconsistent queuing times experienced when using the Volunteer scheduler.

The best results, see Figure 10, were experienced when using the Auction df_load-dif auction scheduler. The columns are vertical, with little horizontal scatter. Essentially, the more vertical the columns (little horizontal variance) the more consistent the queuing times. This indicates that the queuing times at the resources are more consistent and that the load is significantly better distributed to resources than for the other two schedulers. The df_load-diff auction scheduler produced very consistent queuing times and therefore the best distribution of jobs to resources of all schedulers. The graph exhibits compact tightly clustered columns. This stable waiting time is a characteristic shared by most of the auction schedulers and therefore this is a potentially identifiable QoS benefit of using auction based schedulers.

In Figure 11 (a graph of average waiting times) we see that the most stable, i.e. the ones with the lowest amount of deviation (or sensitivity) were the auctions. In terms of the mean waiting time, there is little to choose between the auctions and FairShare.

In summary, the results show that Auction df_load-dif, Auction Random, Auction Load and FairShare all produced comparable results. FairShare had the highest grid utilisation on average, but did not have the shortest simulation time, it also experienced more variability in waiting time. The shortest sim-

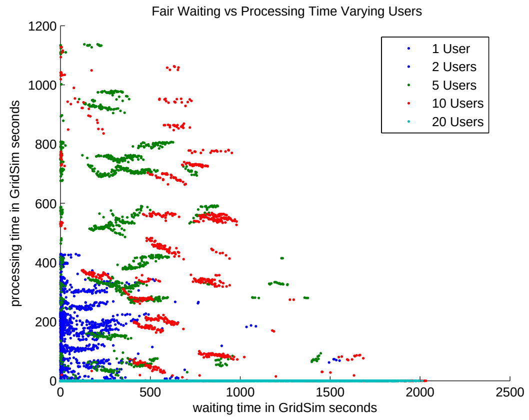

Figure 9: This graph shows the consistency of the queuing times experienced when using the FairShare scheduler. As can be seen in the graph, while the waiting times are highly variable, they do form 3 overlapping clusters increasing in waiting time with the number of users.

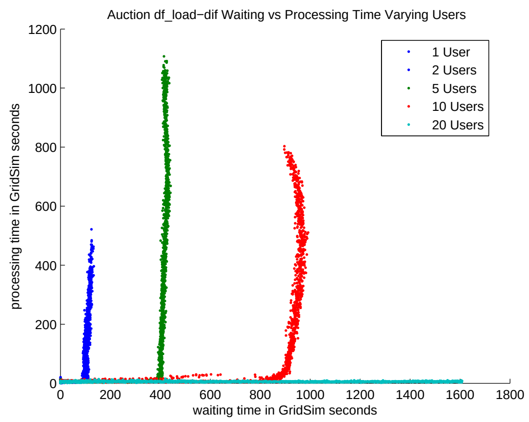

Figure 10: This graph shows the consistency of the queuing times experienced when using the Auction df_load-dif scheduler. The leftmost column is for 2 users, the middle column is for 5 users, the right column is for 10 users, and the points for 20 users lie along the x-axis. All the auction protocols gave similar results.

ulation time was from the Auction Load scheduler. The fact that no mechanism achieved 100% grid utilisation shows that they are not perfect at allocating jobs to resources (something that is possible in the simulation). The auctions produced the most consistent waiting times, although in turn they suffer from a soft start in terms of utilisation.

In Figure 11 (a graph of average waiting times) we see that the most stable, i.e. the ones with the lowest amount of deviation (or sensitivity) were the auctions. In terms of the mean waiting time, there is little to choose between the auctions and FairShare. Volunteer Pooling had the worst performances, due to its uneven loading of resources. Figure 12 shows similar results for the algorithms when varying the number of resources. The results are in general similar to the results in Figure 11 and similar conclusions can be drawn. The only exception to this is FairShare which experiences a far greater deviation in waiting time, yet maintains a similar mean to the auction protocols. This result perhaps reveals some instability in the scheduler.

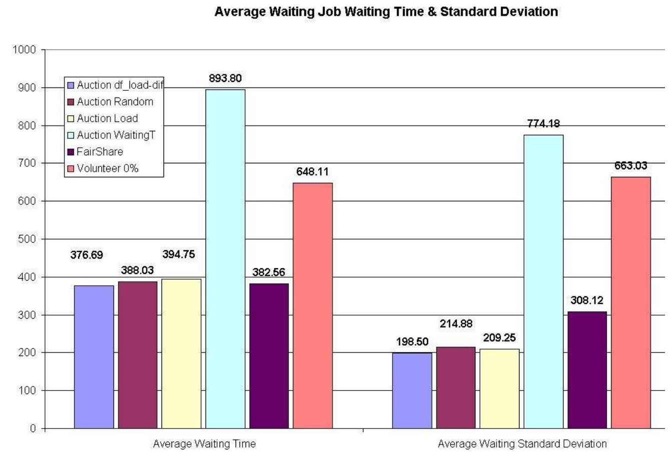

Figure 11: Average Job Waiting Time & Standard Deviation for the scheduling algorithms in this paper when subjected to a variation in the number of users.

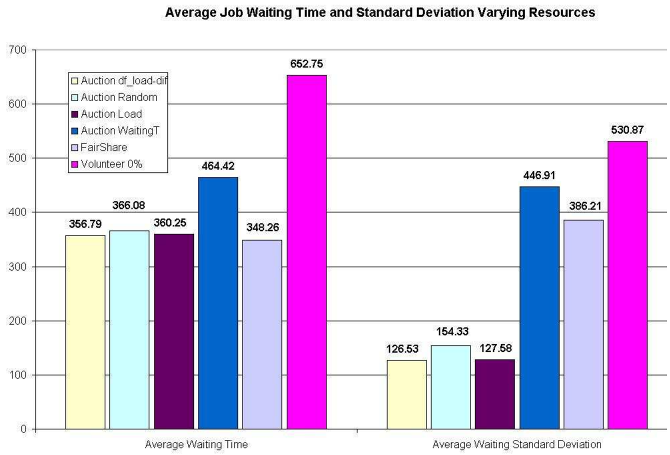

Figure 12: Average Job Waiting Time & Standard Deviation or the scheduling algorithms in this paper when subjected to a variation in the number of resources.

Analyzing the scalability of the mechanisms shows that Volunteer Pooling again performed the worst, when the number of users was increased. However, its performance improved when the number of resources was increased. The auctions were the most stable across variation in number of users and resources, as predicted, while FairShare coped well with increasing the amount of users, but not so well with increasing the amount of resources.

3 Conclusion

Grid resource allocation is a complex problem that has to be tackled one simple step at a time, otherwise the multitude of effects and information is overwhelming. We have set out to characterise different styles of grid allocation, and have turned up some interesting results that are potentially useful to Grid system designers. In general auctions produced a slow start to a batch execution, although their turnaround times were very stable and useful if this were a QoS parameter. This finding alone suggests that auction based resource allocation is best deployed in a continuous allocation scenario. In a burst scenario one of the other allocation mechanisms would return better overall utilisation. Fairshare and other mechanisms that do not have an set-up time (as auctions need) clearly win in overall utilisation with higher utilisation, and less degradation as the number of users increased.

However, as a final conclusion we will state that the utilisation penalty of running an auction is not severe. The implication is that, building an economic allocation system utilising an auction protocol is a reasonable choice, without a large utilisation deficit. In addition the other benefits (scalability, robustness, efficiency, etc.) from using such an allocation protocol will in all likelihood outweigh the loss in utilisation.

4 References

References

Anderson, D. (2004), BOINC: A System for Public-Resource Computing and Storage, *in* '5th IEEE/ACM International Workshop on Grid Computing', pp. 365–372.

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M. & Werthimer, D. (2002), 'SETI@home - An Experiment in Public-Resource Computing', *Communication of the ACM* **45**(11), 56–61.

Anderson, D. P., Korpela, E. & Walton, R. (2005), High-performance task distribution for volunteer computing, *in* 'First IEEE International Conference on e-Science and Grid Technologies, Melbourne, Australia'.

Andreozzi, S., Ferrari, T., Ronchieri, E. & Monforte, S. (2005), Agreement-Based Workload and Resource Management, *in* 'the first international conference on escience and Grid Computing', IEEE, Melbourne, Australia, pp. 181–188.

Bubendorfer, K. & Thomson, W. (2006), Resource managment using untrusted auctioneers in a grid econonmy, *in* 'proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam'.

Buyya, R. (2002), Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD thesis, Monash University, Melbourne, Australia.

Buyya, R., Abramson, D. & Giddy, J. (2000), An Economy Driven Resource Management Architecture for Global Computational Power Grids, *in* 'The 7th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)', Las Vegas, USA.

Buyya, R., Abramson, D. & Giddy, J. (2001), A Case for Economy Grid Architecture for Service Oriented Grid Computing, *in* 'Proceedings of 10th International Parallel and Distributed Processing Symposium: Heterogeneous Computing Workshop', San Francisco, California, USA.

Buyya, R. & Murshed, M. (2002), 'GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing', *Concurrency and Computation: Practice and Experience* **14**.

Casanova, H. (2001), Simgrid: A Toolkit for the Simulation of Application Scheduling, *in* '1st International Symposium on Cluster Computing and the Grid'.

Das, A. & Grosu, D. (2005), Combinatorial Auction-Based Protocols for Resource Allocation in Grids, *in* 'Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium'.

Dimitrakos, T., Randal, D. M., Yuan, F., Gaeta, M., Laria, G., Ritrovato, P., Serhan, B., Wesner, S. & Wulf, K. (2003), An emerging architecture enabling grid based application service provision, *in* 'Seventh International Enterprise Distributed Object Computing Conference (EDOC03), Brisbane, Queensland, Australia'.

Elmroth, E. & Gardfjall, P. (2005), Design and Evaluation of a Decentralized System for Grid-wide Fairshare Scheduling, *in* 'the first international conference on escience and Grid Computing', IEEE, Melbourne, Australia, pp. 221–229.

Feller, M., Foster, I. & Martin, S. (2007), Gt4 gram: A functionality and performance study, *in* 'TeraGrid 07', Madison, WI, USA.

Foster, I. (2002), 'What is the Grid? A Three Point Checklist', Web, http://www.gridtoday.com/02/0722/100136.html.

Foster, I., Kesselman, C. & Tuecke, S. (2001), 'The Anatomy of the Grid - Enabling Scalable Virtual Organizations', *Intl J. Supercomputer Applications* .

Grid Resource Allocation and Agreement Protocol Working Group of the Global Grid Forum (2003), Website, http://forge.gridforum.org/projects/graap-wg.

Howell, F. & McNab, R. (1998), simjava: a discrete event simulation package for java with applications in computer systems modelling, *in* 'First International Conference on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation'.

J.Kay & P.Lauder (1988), 'A fair share scheduler', *Commun. ACM* **31**(1), 44–55.

Kant, U. (2005), Mercatus: A toolkit for the simulation of market-based resource allocation protocols in grids, Master's thesis, WAYNE STATE UNIVERSITY.

Kant, U. & Grosu, D. (2005), Double Auction Protocols for Resource Allocation in Grids, *in* 'Proceedings of the International Conference on Information Technology: Coding and Computing'.

Krawczyk, S. (2006), 'Grid Resource Allocation: an investigation into the viability of economic resource allocation', http://www.mcs.vuw.ac.nz/research/dsrg/sk.pdf.

Malone, T. W., Fikes, R. E., Grant, K. R. & Howard, M. T. (1988), Enterprise: A Market-like Task Scheduler for Distributed Computing Environments, *in* H. B.A, ed., 'The Ecology of Computation', Elsevier Science Publishers (North-Holland), pp. 177–205.

Netto, M. A. S., Bubendorfer, K. & Buyya, R. (2007), Sla-based advance reservations with flexible and adaptive time qos parameters, *in* 'the Fifth International Conference on Service-Oriented Computing', Vienna, Austria.

Parkes, D. C., Kalagnanam, J. & Eso, M. (2001), Achieving Budget-Balance with Vickrey-Based Payment Schemes in Exchanges, *in* 'Proceedings of the International Joint Conference on Artificial Intelligence', pp. 1161–1168.

Pourebrahimi, B., Bertels, K., Kandru, G. & Vassiliadis, S. (2006), Market-based resource allocation in grids, *in* 'second IEEE International Conference on e-Science and Grid Computing'.

R. Wolski and J. Brevik and J. S. Plank and T. Bryan (2003), Grid resource allocation and control using computational economies, *in* F. Berman, G. Fox & A. Hey, eds, 'Grid Computing: Making The Global Infrastructure a Reality', John Wiley & Sons.

Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I. & Wilde, M. (2007), Falkon: Fast and light-weight task execution framework, *in* 'Super Computing', Reno, NV, USA.

Regev, O. & Nisan, N. (1998), The POPCORN market— an online market for computational resources, *in* 'In Proceedings of the First International Conferenceon Information and Computation Economies', ACM Press, Charleston, SC, pp. 148–157.

Sarmenta, L. F. (2001), Volunteer Computing, PhD thesis, MIT Department of Electrical Engineering and Computer Science.

Shannon, R. E. (1975), *Systems Simulation the art and science*, Prentice-Hall.

Sun Microsystems (2007), 'Sun Grid', http://www.network.com/.

Sutherland, L. E. (1968), 'A Futures Market in Computer Time', *Communications of the ACM* **11**(6), 449–451.

Thain, D., Tannenbaum, T. & Livny, M. (2005), 'Distributed computing in practice: the condor experience.', *Concurrency - Practice and Experience* **17**(2-4), 323–356.

Vickrey, W. (1961), 'Counterspeculation, Auctions, and Competitive Sealed Tenders', *The Journal of Finance* **16**(1), 8–37.