A Meta-scheduler with Auction Based Resource Allocation for Global Grids

Saurabh Kumar Garg, Srikumar Venugopal and Rajkumar Buyya
Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{sgarg, srikumar, raj}@csse.unimelb.edu.au

Abstract

As users increasingly require better quality of service from Grids, resource management and scheduling mechanisms have to evolve in order to satisfy competing demands on limited resources. Traditional schedulers for Grids are system centric and favour system performance over increasing user's utility. On the other hand market oriented schedulers are price-based systems that favour users but are based solely on user valuations. This paper proposes a novel meta-scheduler that unifies the advantages of both the systems for benefiting both users and resources. In order to do that, we design a valuation metric for user's applications and computational resources based on multi-criteria requirements of users and resource load. The meta-scheduler maps user applications to suitable distributed resources using a Continuous Double Auction (CDA). Through simulation, we compare our scheduling mechanism against other common mechanisms used by current meta-schedulers. The results show that our meta-scheduler mechanism can satisfy more users than the others while still meeting traditional system-centric performance criteria such as average load and deadline of applications

1. Introduction

Computational resources in large Grids are generally managed by meta-schedulers that interface with the local job schedulers at each resource such as Portable Batch Scheduler (PBS), Load Sharing Facility (LSF) and LoadLeveler, to determine the most appropriate resource for executing a job submitted by a Grid user. Examples of such meta-schedulers include Maui/Moab scheduling suite [1], Condor-G [2], gLite Workload Management System [3] and GridWay [4]. These meta-schedulers mostly focus on improving system-centric performance metrics such as utilization, average load and applications's turnaround time [5].

While the Grids have become more mature with respect to the integration of different components, users have also developed more sophisticated requirements, and are ready to pay upto a certain limit to satisfy them. Current meta-schedulers are unable to satisfy such requirements as they do not consider users' urgency and resource valuation. Thus, we need new scheduling mechanisms which are not only efficient but also that take into account user interests, resource valuation and demand; and schedule user application jobs in a fair manner [13].

In recent years, a number of researchers have proposed economy-based models for more efficient management of Grid resources [6][7][8]. Such models apply well-known and proven economic mechanisms such as markets and auctions to solve the challenges of resource allocations in shared distributed computing environments. Auctions have been particularly preferred by many such projects – for example, Tycoon [7] and Bellagio [8]– as they provide a decentralized structure, are easy to implement, provide immense flexibility to participants to specify their valuations and are considered as the most efficient among current market management systems [9][20]. But these economic-based systems have many limitations. First, while these approaches distribute services fairly, they limit the ability of customers to express fine-grained preferences for services. In addition to that, users may not be able to express their true valuations accurately as they may lack the sophistication to make decisions based on changing resource load and prices. Finally, users with low budgets and urgent requirements may not be able to gain resource allocation as the system may be monopolized by those with large budgets.

This paper presents an auction-based meta-scheduler that aims to overcome the afore-mentioned limitations by taking into account not only the user valuations and resource prices but also other important factors such as resource load and waiting time for the jobs. The meta-scheduler matches jobs to resources using Continuous Double Auction (CDA). The job and resource valuation are then dynamically change depending on the urgency of the job and the load on the resources. We evaluate this mechanism

through extensive simulations using real workload traces and show how the meta-scheduler manages different user requirements in a scenario where the demand for the resources exceeds the supply. Therefore, the main contribution of this paper is the design of a dynamic auction-based meta-scheduling mechanism that internally computes valuation of user applications and resources and performs better than classical scheduling mechanisms in similar conditions.

In the next section, we discuss related scheduling and economy-based resource management projects. Section III presents the system model and details of our scheduling mechanism are presented in Section IV. Sections V presents the experimental setup used for performance evaluation and section VI discusses the results. Finally, we conclude the paper and present future steps in this direction.

2. Related Work

Moab [1] is an advanced meta-scheduler that allows distributed workloads to be run across independent clusters. GridWay [4] is a light-weight meta-scheduler that follows the "greedy approach" to schedule various user applications in round robin manner. However, GridWay does not currently support scheduling mechanisms that schedule a user application considering QoS requirements of other concurrent user applications. gLite Workload Management System (WMS) [3] is part of the EGEE (Enabling Grids for E-Science) toolkit that is used to manage large computing installations extending across hundreds of sites. It uses eager and lazy policies for scheduling jobs to individual resources. Eager scheduling means that a job is bound to a resource as soon as possible and once the decision has been taken, the job is passed to the selected resource for execution. Lazy scheduling waits for a resource to become available before it is matched to the submitted job. The Community Scheduler Framework (CSF) [21] coordinates communications among multiple heterogeneous schedulers that operate at the cluster level. It supports LSF, open PBS and Grid Engine (SGE). In all the above schedulers, while jobs are matched to resources according to system requirements such as memory and number of nodes required, the urgency and priority assigned by the users generally given less prominence. Thus, these schedulers are not suitable for environments with multiple users having different QoS requirements and competing for the same resources as they do not differentiate between users with different requirements. Consequently, researchers have been examining the appropriateness of 'market-inspired' resource management techniques to ensure that users are treated fairly.

Many market-based approaches [6] have been proposed for resource allocation on resources and other distributed systems like Grids. REXEC [13] and Tycoon [9] are proportional share systems in which a task is allocated a share of the resource depending on the proportion of its bid (price) to the total sum of the bids of all tasks executing on that server. LibraSLA [5] prioritizes users on the basis of job deadlines and the user-specified penalties for not meeting them. Bellagio [10] is a system that seeks to allocate resources for distributed computing infrastructures in an economically efficient fashion to maximize aggregate end-user utility. It uses second-price auctions to encourage users to reveal their true valuations of the resources.

Within these systems, users provide job valuations whereas in our mechanism, the meta-scheduler computes its own valuations for user applications based on user input and system conditions. The Valuations are invisible to users or resources. Also, REXEC, Tycoon and LibraSLA primarily aim to improve the profitability and utilisation of the resource providers while our meta-scheduler aims to benefit both the users and the resources.

The auction-based mechanisms have been the subject of many previous studies. Grosu, et al. [14] compare resource allocation protocols using First-Price, Second-Price Vickery and Double Auctions (DA). They show that DA favors both users and resources while First-price Auction is biased towards resources and Vickery auction favors users. Gomoluch, et al. [15] compared CDA with Proportional Share Protocol for resource allocation and concluded that CDA performs better in most of cases. Kant, et al. [16] compared three different Double Auction protocols and concluded that the CDA protocol performs better than the others in terms of resource utilization, resource profit and spent budget. Pourebrahimi, et al. [17] used CDA for resource allocation on grids that employed a pricing mechanism based on historical data. Therefore, we have opted for CDA as the basic mechanism for our meta-scheduler. Our meta-scheduling environment is closest to that of GridWay [4] but we have used CDA-based mechanism to allocate resources to users.

Our work is quite different from other market-based systems as the auction mechanism is invisible to both the users and the resources. Moreover, our meta-scheduler assigns values to applications and resources based on dynamic conditions such as resource load, demand and supply of resources, and deadline of user applications.

3. The Meta-Scheduling System

The meta-scheduler presented follows the model commonly found in large computing installations across educational and research institutions [18]. In this model, resources are managed at different sites by administrators (Service Providers) who have to cater to the user's needs at their site. Batch scheduling systems that manage these resources are generally organised as a collection of user-accessible job queues where a queue may allow submission of only those jobs that meet certain criteria (e.g. within

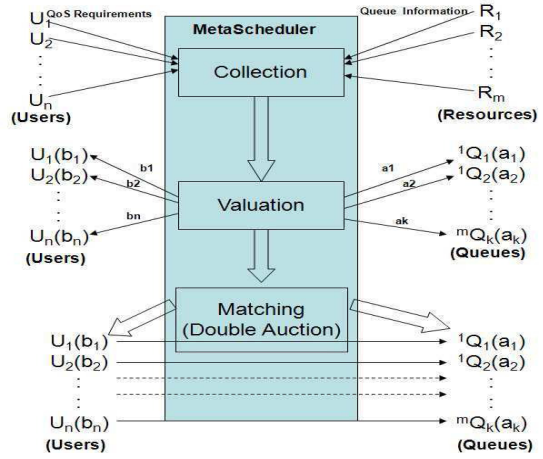


Figure 1. The DAM's mechanism

a maximum job size) [19]. Providers assign queues for exclusive use of the meta-scheduler, and supply information about load and waiting times of each such queue to the meta-scheduler at regular intervals. Providers also supply an initial valuation (cost) to meta-scheduler for running a job in a queue based on their estimation of the relative strength of their resources. Users submit their applications to the meta-scheduler for execution within some deadline at the resources in the computing installation/Grid.The deadline is estimated by a user on the basis of expected execution time of the application and his/her urgency. The user's application is valued on the basis of the result urgency and the information provided by the meta-scheduler, about the costs of using the resources at regular time-intervals. In the current system, we assume user applications to follow the Bag-of-Tasks (BoT) model, that is, the tasks within the application are translated to independent jobs on the Grid resources [22]. The meta-scheduler uses the information supplied by the providers and the users to match jobs to the appropriate queues on the resources. The meta-scheduler calculates its own valuation for the resources and user applications.The meta-scheduler acts as an auctioneer that matches jobs to resources using a CDA mechanism. The objective of the meta-scheduler is to conduct the resource allocation fairly so that the maximum number of applications is completed and the QoS requirements of users are met. It also aims to distribute the load fairly across the different resources. Figure 1 shows the elements of the meta-scheduler, which can be divided into three parts: (1) collection: meta-scheduler collects queue informations, (2) valuation: assign values to the user applications and resource queues, and finally, (3) matching using CDA. In the Figure 1, U_n represents user application, a_k and b_n represent ask and bid, and mQ_k represents resource queue. At regular intervals (or scheduling intervals), the meta-scheduler matches the jobs to the resource queues if the deadline constraint of the application is satisfied. If a job cannot be matched, then it is considered in the next scheduling interval.

3.1. Continuous Double Auction (CDA)

In a CDA, both sellers and buyers submit bids to an auctioneer who continually ranks them from highest to lowest in order to generate demand and supply profiles. From the profiles, the maximum quantity exchanged can be determined by matching selling offers or asks, starting with lowest price and moving up, with the demand bids, starting with highest price and moving down. This format allows buyers to make offers and sellers to accept those offers at any particular moment.

An auctioneer clears asks and bids continuously as they arrive. When it receives a new bid b_i, it searches for an ask a_j lesser than bid b_i. If it finds one, it sends the match to the bidder with a trading price that is generally the average of the bid and the ask values. Otherwise,the bid is rejected. When it receives a new ask a_i, it searches for a bid b_j greater than a_i. If it finds such a bid, it sends the match to the supplier, otherwise the ask will be inserted in the ask list.

We have used a slightly different version of this double auction mechanism in our scheduler mechanism. The CDA is held at regular scheduling intervals and all the bids are matched at the end of these intervals.Within our meta-scheduler job valuation is considered as a bid while resource valuation is considered as an ask.

3.2. Pricing Mechanism

The Grid services may be priced based on the cost of infrastructure, and economic factors like supply and demand. However, user needs and urgency, and simultaneously, efficient utilization of Grid services must be reflected through pricing (valuation) of user applications and resources. Therefore, the meta-scheduler must generate a pricing metric for both users and resources that takes into account all these constraints. This pricing is dynamic, that is, in each scheduling cycle; it gets updated based on various parameters, and the dynamic demand and supply of system.

Valuation (Pricing) of Resources: In order to balance load across independent grid services, the meta-scheduler tries to submit more jobs to the least loaded resources. Also, the most urgent job must be matched to the fastest queue. Therefore, the valuation of resources should be such that

the resource with minimum load should get minimum value (as in CDA, the maximum bid is matched to minimum ask). Therefore, $P_R(t)$, the price of a resource at time, is determined by the following:

$P_R(t) \propto w_{R(t-1)}$, where $w_{R(t-1)}$ is average queue waiting time,
$P_R(t) \propto \frac{Demand}{Supply}$,
$P_R(t) \propto c_R$, where c_R is initial price given by the resource,
$P_R(t) \propto l_{(t-1)}$, where is $l_{(t-1)}$ load of resource,

After combining above equations, we get the price metric for the clusters as:

$$P_R(t) = K \times w_{R(t-1)} \times c_R \times l_{(t-1)} \quad (1)$$

where K is a proportionality constant.

Algorithm 1: Pseudo code for DAM

```
1  ScheduleList← null;
2  while current_time ¡ next_schedule_time do
3      RecvResourcePublish(Pj)//from providers
4      RecvJobQos(Qj) //users
5  endw
6  CalculateDemand (Qj)
7  CalculateSupply(Pj)
8  UpdateBidPrices(Qj)
9  UpdateAskPrices(Pj)
10 List asks←Sort_Asks(Pj)
11 List bids←Sort_Bids(Qj)
12 Let j=0 //pointer to resource list a?asks(j)
13 n=availableQueueSlots(a)
14 foreach bid i in list bids do
15     b←bids(i) foreach task j of bids b do
16         if n¿0 then
17             if a.value()¡b.value then
18                 if check_Deadline(i,a) then
19                     Schedule j=AssignResource(i,a)
20                     addinSchedulelist(Schd_List)
21                     n–
22                 else
23                     Goto line 37
24                 endif
25             else
26                 if IsEmpty(asks) then
27                     break
28                 else
29                     j++
30                     a←asks(j)
31                 endif
32             endif
33         else
34         endif
35     endfch
36 endfch
37 foreach element in Schd_List do
38     notifyuser()
39 endfch
```

Valuation (Pricing) of User Application: As discussed previously, each user submits to the meta-scheduler his/her budget (b_u), deadline (d_u), application length (l_u), and the number of nodes required (n_u). Let $P_u(t)$ be the valuation of the user application. As user applications with the maximum budget must be given higher priority,

$$P_u(t) \propto b_u$$

Also, the more urgent a job, higher its priority, therefore,

$$P_u(t) \propto \frac{1}{d_u - T_t}$$

where T_t is the current time. A user application that has waited longer gains higher priority.
$P_u(t) \propto T_t - S_t$, where $S_t$is the time the application was submitted.

Finally, as the valuation is based on the demand and supply of resources (clusters) in the system,
$P_u(t) \propto \frac{Demand}{Supply}$, where $Demand$ is total number of task to allocate and $Supply$ is the total number of CPUs in all resources.

Therefore, we get the following metric for pricing of user applications,

$$P_u(t) = k_u \times b_u \times \frac{1}{d_u - T_t} \times \frac{Demand}{Supply} \times (T_t - S_t) \quad (2)$$

3.3. DAM: Double Auction-based Meta-scheduling

DAM algorithm uses the valuation metric given in Equations (1) and (2) to assign valuation to user applications and grid resource queues at the end of every scheduling interval. These valuations are used to match each component job of the user application to suitable resource queues. Algorithm 1 shows the scheduling mechanism in the meta-scheduler. The meta-scheduler schedules the user applications on independent resources at discrete time intervals (Line 2). In each interval, the meta-scheduler waits for user's request for resources (Line 4) and the information about the waiting times in each of the resource queues supplied by the corresponding provider (Line 3). At the end of scheduler interval, the meta-scheduler calculates the demand for resources and their supply (Line 6-7). The meta-scheduler uses supply and demand information to assign values to user applications (bids) and to each resource's queue(asks) using the pricing mechanisms presented in the previous section (Line 8-9). Then, all asks are sorted in ascending order (Line 10) and all bids are sorted in descending order (Line 11). For each bid (or a job in a user application), the meta-scheduler finds an ask (or a resource queue) of a lesser price than the valuation (Lines 18-19). It then checks that the deadline for the application is not violated by the assignment (Line 20).

If so, the job is then assigned to a slot in that queue (Line 21) and this assignment is added to the schedule list (Line 22). If the deadline check fails, then, due to the manner in which the pricing was performed, no other ask can be matched to the job's bid (Line 25). Therefore, the job is removed from the bids list for that scheduling interval.

Complexity of meta-scheduling mechanism and matching: Let n be the number of user applications with total N tasks. Let m be the number of resources each containing q number of queues.The main operations performed during meta-scheduling are:

1. The calculation of demand and supply which are of order *O(qm+n).*
2. The valuation of user application is of order *O(n)*
3. The valuation of cluster's queue is *O(mq)*
4. The sorting of asks and bids is of *O(n*log*n+mq*log*(mq))*
5. Matching in worst case is of order *O(N)* when all tasks are matched to corresponding queue slots.

Therefore, the resultant complexity of the meta-scheduling mechanism is summation of time order of above operations i.e.

$$[O(qm+n)+O(n)+O(mq)+O(n\log n+mq\log(mq)) \\ +O(N)] = O(n\log n+mq\log(mq)+N)$$

4. Experimental Configuration

We use GridSim [23] to simulate our double auction-based mechanism for matching user applications with Grid resources. Our experiments employ real workload traces gathered from existing supercomputers and collected in the Parallel Workload Archive [24]. For this experiment, we have selected a subset of 500 applications (associated with each user) from the trace of the Linux cluster (Thunder) at LLNL for the duration between February and June 2007 [24]. Since the workload trace does not contain any information about the user's deadline and budget, these were generated using a uniform random distribution. For a user application with a runtime r, the deadline was generated randomly with uniform distribution between r and 3r. The average initial budget given by the user varies between 90000 and 160000 currency units. The user budgets are assigned so that at least half of users can afford to execute their application on the resources with highest valuation. The user's application's runtime, number of processing elements (PEs) required and submission times are taken from the workload traces. The user application is modelled as a Bag-of-Task application, consisting a set of independent jobs, the size of which is the same as the number of processors required by the application. Therefore, a total of 30,000 jobs were simulated through these experiments. The computing installation modelled in our simulation is that of a subset of the European Data Grid 1 test bed that contains 8 Grid resources spread across six countries connected via high capacity network links. The configurations assigned to the resources in the testbed for the simulation are listed in Table 1. The configuration of each resource is decided so that the modelled test bed would reflect the heterogeneity of platforms and capabilities that is normally the characteristic of such installations. All the resources were simulated as clusters of PEs that employed easy backfilling policies in order to improve responsiveness. The PEs associated with each cluster in Table 1 are allocated exclusively to the meta-scheduler. It can be presumed that each cluster has a separate set of PEs assigned exclusively to local users. We have sub-divided the allocated PEs of each cluster into 3 queues in ratio of 1:2:3 of the total number of PEs in the cluster. Within these experiments, the clusters are dedicated to Grid users, and therefore, do not have any load from local users. The processing capabilities of the processors were rated in terms of Million Instructions per sec (MIPS) so that the application requirements can be modeled in Million Instructions (MI). The average initial valuations that are assigned to each resource is between 4.5 and 9.5 currency units per processor per second. We have compared our scheduling mechanism

Table 1. Simulated EDG Testbed Resources

Resource name (location)	Number. of nodes	Single PE rating (MIPS)
RAL(UK)	2050	1140
Imperial College(UK)	2600	1330
NorduGrid (Norway)	650	1176
NIKHEF (Netherlands)	540	1166
Lyon (France)	600	1320
Milano (Italy)	350	1000
Catania (Italy)	200	1330
Padova (Italy)	250	1200

to four other mechanisms for these experiments:

- **Shortest Job First (SJF):** In this mechanism, the jobs are prioritized on the basis of estimated runtime. This is very common algorithm used by resource management systems.
- **First Come First Serve (FCFS):** A job is assigned to the first available queue. This is one of the scheduling algorithms used in meta-schedulers like GridWay[4].

- **Highest Budget to Fastest Queue (HBFQ):** In this mechanism, the job with the maximum budget is assigned to the queue with minimum waiting time. This mechanism favours applications with large budgets.
- **FairShare or Proportional Share:** In this mechanism, in each schedule interval, each application is assigned queue slots proportional to the ratio of its budget to the combined budget of all the applications. This market-based algorithm is used in REXEC [13].

We tested our meta-scheduling mechanism by performing a series of experiments that compare our mechanism with the others. The following performance metrics are used to compare fairness and user satisfaction of our mechanism with others:

- Deadline urgency: Deadline urgency (u), which indicates user urgency to get his application completed, is defined as:

$$u = \frac{deadline - starttime}{executiontime} - 1 \qquad (3)$$

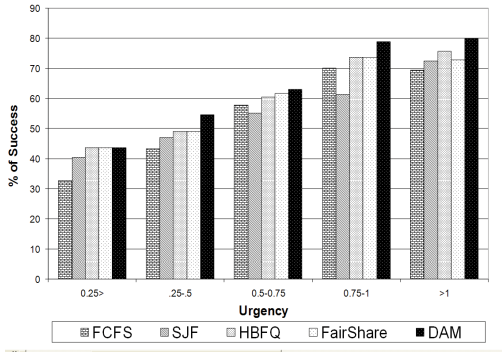
  where $starttime$ is start time of the user application and $executiontime$ is the execution time of user's application. The deadline is considered very urgent when $u < 0.25$, urgent when $0.25 < u < 0.5$, intermediate when $0.5 < u < 0.75$, relaxed when $1 > u > 0.75$ and very relaxed when u¿1. This metric shows how the scheduler deals with users with different demand on time.
- Budget per job: The budget (valuation) provided by the user for his/her application is divided by the number of jobs contained within the application to normalize the budget across all the applications. This metric examines how the schedulers allocate resources fairly among different users with different budget groups.
- Number of Deadlines missed with increase in number of user applications: This metric is used to examine how the scheduling algorithms are able to cope up with multiple users when demand for resources exceeds their supply.

To evaluate the benefit of our meta-scheduling mechanism for the resources we have also considered the following metrics:

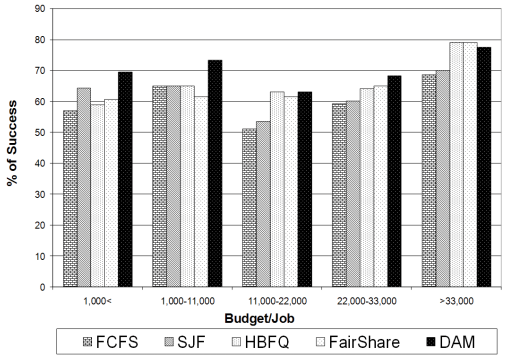
- Average Load of a Resource: This is the number of processors occupied across all the queues available to the meta-scheduler in a resource, divided by the total number of processors allocated to these queues.
- Average Valuation of a Resource: This is the average of the valuations assigned to the queues in a resource by the meta-scheduler.

5. Analysis of Results

5.1. Benefit for the users



(a) Effect of user urgency



(b) Effect of user budget

Figure 2. User Benefit.

This section shows how our meta-scheduler is more fair to users by not only completing the most number of applications with different QoS needs but also benefiting every user in different urgency and budget groups.

Effect of Deadline urgency . Figure 2 shows the percent of all user applications successfully completed based on the urgency of user application deadlines. An application is not considered successfully completed until all its component jobs are executed. Figure 2 clearly shows that DAM has outperformed other meta-scheduling mechanisms by completing more applications in every urgency group. This is because the valuation of the user application increases with the urgency in DAM but remains static in other mechanisms. The longer an application's jobs wait to be allocated, the higher their value. In this way, the resource allocation is fair to all the applications. FCFS has performed the worst in

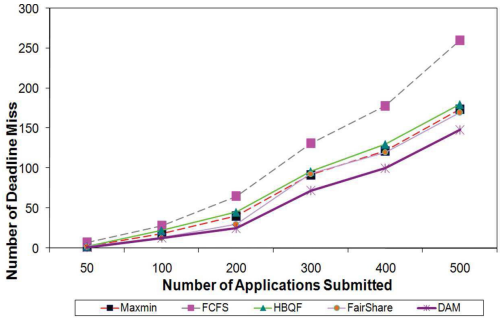


Figure 3. Number of Deadline Miss

every case. The reason for this behavior is that FCFS mechanism may result in jobs with relaxed deadline being scheduled earlier than those with strict deadlines. In the case of DAM, it is able to delay users with relaxed deadlines so that it can accommodate applications with strict deadline.

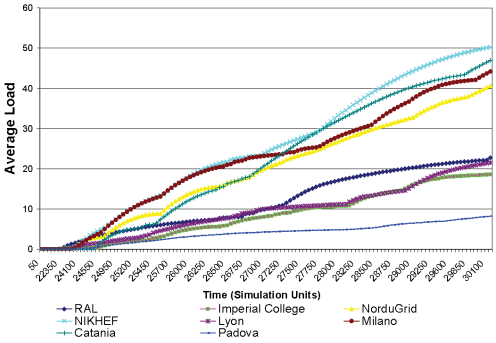
Effect of User Budget . From Figure 3, we can see that DAM completes at least as many as user's applications as FairShare. The FairShare fairly allocates resources among various users on the basis of user budgets. In the case of DAM, difference between the minimum and the maximum percentage of successfully completed is about 14% while for HBFQ, this difference is the highest a mong all other mechanism that is 20%. It can be inferred that HBFQ is least fair for all users while DAM scheduling almost equal number of users from all budget groups. This is due to the fact that our valuation of user application is not evaluated just on the basis of budget given by user as in HBFQ. Thus unlike DAM, HBFQ and FairShare provide more resources to applications with higher budgets without considering the deadlines of other users.

Number of Deadline miss . From Figure 4, we can clearly see as the demand for resources (number of user applications) increases, the number of applications that missed their deadline also correspondingly increases due to the scarcity of resources. In this scenario, DAM is able to satisfy more number of users than other mechanisms as DAM is assigning valuation to user applications according to deadline. In FCFS, deadline misses is increasing rapidly due to starvation of many urgent applications.

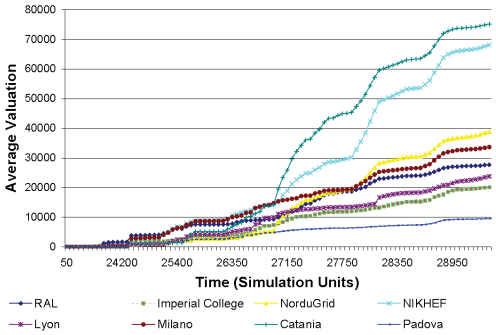
5.2. Benefit for Resources

In this section, simulation results show how DAM affects the load on different resources.

Variation of Weighted Load and Valuation of Resource with Time . Using our DAM, Figure 5 shows that the



(a) Load Variation



(b) Valuation Variation

Figure 4. Resource Benefit.

load on all the resources increases over time as the number of user applications increases, while Figure 6 shows that the valuation of these resources also increases. This is due to our valuation metric increasing the valuation of a resource when its load increases. This in turn curbs the demand for the resource since our DAM assigns jobs to resources with the lowest valuation first. Hence, jobs are being transferred to other resources. Consequently, the valuation of the least-loaded Padova resource is also the minimum.

However, Figure 6 shows that there is a tremendous increase in the valuation of the Catania and NIKHEF resources. As listed in Table 1, even though Catania has the least number of PEs, it is still being assigned jobs since it has the highest PE rating and thus requires shorter average waiting time for each of its queue as compared to other resources. In case of NIKHEF resources, the reason for high valuation is not only very high load but also job submitted are larger in size which increased the queue waiting time further. Thus resultant increased in the waiting time lead to more high valuation of NIKHEF resources. On the other hand, the RAL and Imperial College resources with the highest number of PEs are among the least-loaded resources and hence also valued least.

6. Conclusion

In this paper, a double auction based meta-scheduler is presented. The meta-scheduler uses valuation metrics to map user applications to resources consisting of independent resources in a fair and efficient manner to benefit both user and resource side. The valuation metrics are designed to assign values to user applications on the basis of user QoS requirements and to resources on the basis of their load. This valuation is dynamic in nature and changes as the load of resources increase or decrease. The valuation of user application also varies depending on the length of time the application waited in the meta-scheduler to have all its component jobs executed on matching resources. We also compared our scheduling mechanism (DAM) with two traditional scheduling mechanisms (SJF and FCFS) and two market-based scheduling mechanisms (HBFQ and FairShare). Our results clearly show that our mechanism is not only able to satisfy more users, but also benefited all users in different urgency and budget ranges. Moreover, it is shown that our valuation metric can be very effective in balancing the load across various resources. Thus in overall our mechanism benefited both the resources and users. In our current design, every task of an application has same value and are considered equivalent. But, some tasks are more critical or resource intensive than others. In the future, we would like to develop differential pricing mechanisms for such scenarios and apply them to our meta-scheduler. From a resource perspective, we have assigned equal prices to all the queues within a resource. It would be interesting to differentiate queues based on their relative capabilities.

Acknowledgements

We would like to thank our colleagues - Sungjin Choi, Marco Netto and Chee Shin Yeo - for their comments and suggestions on this paper. This research is funded by the International Science Linkage grant provided by the Department of Innovation, Industry, Science and Research (DIISR) and Australia Research Council (ARC).

References

[1] Supercluster Research and Development Group. Maui Scheduler Version 3.2 Administrator's Guide, 2004.

[2] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *10th IEEE Int. Sym. on High Performance Distributed Computing*, San Francisco, Aug. 2001.

[3] Glite Workload Manager, http://glite.web.cern.ch

[4] E. Huedo, R. Montero, & I. Llorente, "A framework for adaptive execution on grids",*Software - Practice and Experience*, vol. 34, no. 7, pp. 631-651, June 2004

[5] Chee Shin Yeo and Rajkumar Buyya, "Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility," *Proc. of the 7th Int. Conf. on Cluster Computing*, Boston, Sept. 2005.

[6] R. Buyya, D. Abramson, & S. Venugopal, "The Grid Economy", *Proc. of the IEEE*, Vol. 93,pp. 698-714, Mar. 2005.

[7] K. Lai , L. Rasmusson, E. Adar, L. Zhang,& A. Huberman,"Tycoon: An implementation of a distributed, market-based resource allocation system",*Multiagent Grid System*, Vol. 1, No. 3, pp. 169-182, Aug. 2005

[8] A. AuYoung, B. Chun, A. Snoeren, & A. Vahdat, "Resource allocation in federated distributed computing infrastructures", *Proc. of the 1st Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure*,Oct. 2004.

[9] R.B. Myerson & M.A. Satterthwaite, "Efficient Mechanisms for Bilateral Trade," *J. Economic Theory*, vol. 29, no. 2, pp. 265-281.April 1983

[10] Platform Computing. LSF Ver. 4.1 Admin.'s Guide, 2001.

[11] IBM. LoadLeveler for AIX 5L Ver. 3.2 Using and Administering, Oct. 2003.

[12] Sun Microsystems. Sun ONE Grid Engine, Administration and User's Guide, Oct. 2002.

[13] B. N. Chun & D. E. Culler, "User-centric performance analysis of market-based cluster batch schedulers,*2nd IEEE Int. Sym. on Cluster Computing and the Grid*, May 2002.

[14] D. Grosu and A. Das, "Auction-based resource allocation protocols in grid", *Proc. of the 16th Int. Conf. on Parallel and Distributed Computing and Systems*, Nov. 2004.

[15] J. Gomoluch & M. Schroeder. Market-based resource allocation for grid computing: A model and simulation, 2003.

[16] U. Kant & D. Grosu, "Double auction protocols for resource allocation in grids", *Proc. of the Int. Conf. on Information Technology: Coding and Computing*, 2005.

[17] B. Pourebrahimi, K. Bertels, G. Kandru,& S. Vassiliadis, "Market-based resource allocation in grids", *Proc. of 2nd IEEE Int. Conf. on e-Science and Grid Computing*, 2006.

[18] Gridway Installation, Available:http://irisgrid.rediris.es.

[19] G. Pacifici, M. Spreitzer, A. Tantawi, & A. Youssef, "Performance Management for Web Services", Research Report RC22676, IBM T. J. Watson Research, 2003

[20] D. Friedman & J. Rust, eds., "The Double Auction Market: Institutions, Theories, and Evidence," Addison-Wesley-Longman, Mass, 1993.

[21] C. Smith, "Open source metascheduling for virtual organizations with the community scheduler framework (CSF)", Technical whitepaper, Platform Computing, 2003.

[22] A. Iosup, M. Jan, O. Sonmez, & D.H.J. Epema, "The Characteristics and Performance of Groups of Jobs in Grids", *Euro-Par 2007*, August 2007

[23] R. Buyya and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing". *Concurrency and Computation: Practice and Experience*, Dec. 2002.

[24] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload,May 2005.