Proceedings Paper:
Kavanagh, R and Djemame, K (2012) The ISQoS Grid Broker for Temporal and Budget Guarantees. In: Vanmechelen, K, Altmann, J and Rana, O, (eds.) Economics of Grids, Clouds, Systems, and Services. GECON 2012, 27-28 Nov 2012, Berlin, Germany. Lecture notes in computer science, 7714 . Springer , 76 - 90. ISBN 978-3-642-35193-8

https://doi.org/10.1007/978-3-642-35194-5_6

The ISQoS Grid Broker for Temporal and Budget Guarantees

Richard Kavanagh and Karim Djemame

School of Computing, University of Leeds,
Leeds, LS2 9JT, UK
{screk,k.djemame}@leeds.ac.uk
http://www.comp.leeds.ac.uk

Abstract. *We introduce our Grid broker that uses SLAs in job submission with the aim of ensuring jobs are computed on time and on budget. We demonstrate our broker's ability to perform negotiation and to select preferentially higher priority jobs, in a tender market and discuss the architecture that makes this possible. We additionally show the effects of rescheduling and how careful consideration is required in order to avoid price instability. We therefore make recommendations upon how to maintain this stability, given rescheduling.*

Keywords: Time-Cost Constrained Brokering, SLA, Negotiation, Job Admission Control, Grid

1 Introduction

Grids enable the execution of applications in a distributed fashion. It is however, common that such applications are served in a best effort approach only, with no guarantees placed upon the service quality. This is primarily due to the emphasis in production Grids being placed upon the queuing of jobs ready for computation, with the sole intent of maintaining high resource utilisation, rather than user satisfaction. It has also been known for some time that guaranteed provision of reliable, transparent and quality of service (QoS) oriented resources is the next important step for Grid systems [23, 3].

In many commercial and scientific settings, guarantees that computation is going to be completed on time are required. It is therefore important to establish during the submission of a job the requirements of the users in terms of completion time and the priority they hold for the work.

In order to motivate our work and illustrate the need for time guarantees we present two scenarios. The first is a commercial scenario, such as animation, where frames may be computed overnight before the animation team arrives, partial completion of the work delays or stops the team from starting the next days work [2]. The second scenario is in an academic environment where it is common before conferences for Grids to become overloaded [16]. To further focus our work and enhance its relevence we consider the types of application

running upon the Grid. The focus of our work is hence upon Bag of Task based applications, which are the predominate form of workload in Grids [16].

Given the finite resources available (including budget), it is also wise to prioritise jobs based upon the importance to the end user. In order that this prioritisation is provided correctly an economic approach is used to ensure users give more truthful indications of their priorities [6, 21].

In delivering these time and cost requirements, it presents the added advantage that Grids can be moved away from the best-effort service which limits their importance, as users' would be more willing to pay or contribute resources if results are returned on time, as late results are of limited use[20].

The main contributions of this paper are to:

- Introduce the architecture of the Intelligent Scheduling for Quality of Service (ISQoS) broker. Its focus is upon job submission via the formation of service level agreements, which cover job completion time and cost.
- Demonstrate how a tender market can be formed that actively selects jobs of a higher priority, via their economic properties. We show this as part of a transition from the dominance of the temporal constraints to the budget constraints of a job submission.
- Introduce rescheduling and demonstrate how this can lead to price instability, which is a key factor in a successful Grid market [34]. We therefore make recommendations to counteract this instability.

The remaining part of the paper's structure is as follows, in section 2 We discuss the ISQoS general architecture, which leads on to a more in-depth discussion of the broker's service pricing mechanism and offer evaluation in section 2.2. The pricing of resource and scheduling mechanisms used are then covered in section 2.3, which are the main aspects of this paper's experimentation. The setup for this experimentation can be found in section 3 and the results may be found in section 4. We then discuss related work in section 5 and conclude in 6.

2 The Broker

In this section we discuss the overall architecture of the ISQoS Broker. This broker is primarily aimed at parameter sweep / bag of task applications [8], which are the predominant workload upon the Grid [16].

It aims towards providing QoS via economic mechanisms and the ability to negotiate to find the provider that is most likely to be able to complete the computation on time and on budget.

The ISQoS Grid architecture is service oriented and is WS-Agreement for Java (WSAG4J) [15] based and presents services to execute jobs. Avoiding classical architectures such as Globus [12] has allowed the focus upon scheduling and the ability to negotiate service level agreements (SLAs). This includes the capacity to perform scheduling for indicative purposes only, i.e. not for committing to work, but merely to generate a candidate schedule for negotiation purposes. The

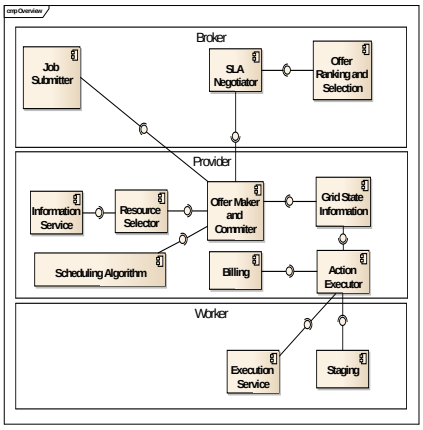

Fig. 1: Overview of Architecture

use of scheduling in the architecture also brings it closer in style to the Maui Scheduler [18] and OpenCCS [3] opposed to queuing based methods.

The broker focuses upon standards such as JSDL [27], but has the capacity to swap out this term language used to describe the work being performed, so long as it is XML based so that it can form part of the agreement. This is achieved by having the part of the XML describing jobs contain an `xsd:any type` for attaching the term language describing the job. This flexibility is also present in the information provider which can use either GLUE [5] or Ganglia [13] based representations of resources.

2.1 Broker Architecture

We present the overview of our architecture in Figure 1. It primarily consists of three tiers: A broker, a provider and workers. The broker selects the provider to use while the provider commands the workers to execute Grid jobs.

Jobs are submitted via the job submitter of the broker to the offer maker and committer component of the provider. The broker is expected to contact various different resource providers in order to establish a competitive market place.

The information submitted includes the job's *resource requirements*, a *budget* that the user has assigned for the completion of a job, the amount of *markup* the broker makes for the service, which can be used as a notion of priority and finally its temporal requirements. These will be shown as a *due date* by which the job should be completed by and a *deadline* by when it must be completed.

The resource providers perform initial scheduling of the tasks within a job so they can derive an estimate for completion time and price. They achieve this by using a scheduling algorithm that is plugged into the provider. The price is based upon the provider's local pricing mechanism which feeds information into the scheduling algorithm. The schedules generated are then converted to offers to complete the work, which are then submitted back to the broker. These offers

include information about the time and cost for completing the job and the time for completing each task within a job.

The broker from the offers that have been returned, ranks them and filters out poor offers i.e. sort by earliest and filter out unprofitable offers, though various selection mechanisms may be used. One such example is establishing a going rate from historic offers and comparing the current offer to that rate [19]. The broker then asks the user if it acceptable to proceed with the job at a given price and completion time.

If the offer made to the user is acceptable the broker then submits the bag of tasks that make up the job to the winning provider. In terms of the experimentation in section 4 then any job that has its requirements met is accepted. In the case of a production Grid, the user could resubmit the job with different budget and time requirements in order to insure the job is accepted.

An accepted job is placed into the schedule and is recorded in the current state of the Grid. The state records the mappings between workers and their jobs as well as other information such as the current resource and job statuses. The workers are represented as objects which maintain a copy of the XML description of the resource and a reference to the XML parser used to interpret the description of the job. This XML description of the job is taken from the agreement which is attached as an `xsd:any type`, which allows for XML other than JSDL documents to be used. The parser maintains a set of default questions which can be asked about the job i.e. what is the size of the memory available? or the what is the CPU speed?

The descriptions of the tasks are treated in a similar fashion and a basic list of default questions are provided i.e. how much memory is needed. The resource selector for a given task can then be asked to provide the list of acceptable workers to the scheduler, without it needing to have a great understanding of what a resource entails.

In terms of a job's breakdown a job is a collection of tasks in the bag, that need executing. Each task has several actions, these equate to: stage in, execute, stage-out and clean the worker. Stage-out merely holds to the meaning of transferring data away from the worker. Clean removes the task's working area upon a worker node. If the Clean event is the very last action to be performed then the job is submitted for billing.

The action executor merely waits for the next action in the schedule to be ready for execution. At this point it sends a signal to the worker nodes to begin their work.

2.2 Job Pricing and Offer Evaluation

The generation of offers and pricing follows work in [19]. It is hence summarised here, to aid the discussion of our work. From each of the providers a resource cost and completion time for the work is obtained. The broker needs to make a profit so must make a mark-up from this initial resource cost. This is done as a percentage of the original resource cost as shown in Figure 2a. In order for a budget violation not to occur this must be below the maximum budget.

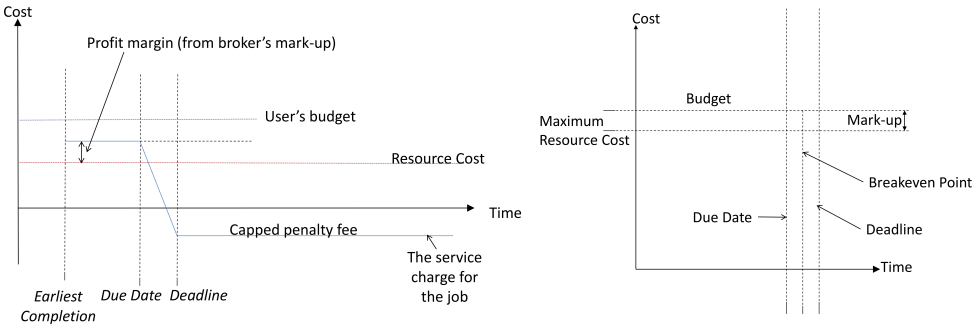

Fig. 2: The generation of offers

This gives rise to a notion of *budget slack* or spare budget available. It also gives rise to *budget resilience* i.e. the difference between resource cost and the budget available, as the broker could notionally decide to use some of its profit towards making a job complete on time. Hence the profit the broker makes represents the minimum amount of money available, should it be required.

If a job is returned on time then it achieves its full service price. If it is delayed then it returns a fraction of it based upon how far the completion time is between the due date and deadline. Instead of having a penalty to the end user the service charge is stopped at 0 and the broker pays for any resources used that did not fail. This has useful properties in that the breakeven point becomes a fixed point between the due date and deadline [19]. In doing so the offer market is generated which is shown in Figure 2b.

2.3 Scheduling and Resource Pricing

The scheduling algorithm in use may be swapped out at will, as can the mechanism used to select a price for the resources. The scheduling algorithms that have been implemented and used in this paper are round robin and a variation of round robin that performs rescheduling. This variation is given below:

```
FOR EACH (Task) {
  Get next worker in round robin order that meets resource
requirements
  IF Worker.isEmpty() {
    Place Task's Actions; BREAK; }
  FOR EACH (ACTION in Task) {
    FIND insertion point WHERE (
    Later Actions will not go past their Due Date
    AND No action already started will be moved
    AND Any action due to start will not be moved ) {
```

```
            Place Action in earliest position possible; BREAK;
        }
    }
```

The rescheduling variant takes the selected workers actions and moves them along as far as their due dates will allow and places new actions in before them. The aim is to have all new jobs start as early as possible and to reduce the dependence upon arrival order in terms of how well a job is served. Actions may not be moved however, if they are due to start or have already started and movement of actions should also not make an action go past its due date.

The resource pricing mechanisms implemented aim to dynamically map the demand for resources to the price. In this paper time for both network and resource usage are billed equally. The charge per second of worker time derives from the count of actions that the provider currently has in its schedule, which is then mapped to a price. This although simple, as discussed in section 3, should be suitable for experimentation.

3 Experimental Setup

We perform experimentation with the aim of demonstrating the effects of temporal and budget constraints upon machine/task selection, with the intent of creating a mechanism by which job prioritisation may take place. We focus this experimentation upon high load scenarios where correct selection is most required. The configuration of the experiments performed is described next.

We sent 100 jobs with 8 tasks each into a Grid with 2 providers. Each provider had 4 virtual machines, of which one also acted as a head node. Jobs were submitted with a 30 second gap between submissions, from a separate broker virtual machine instance. This is shorter than the time it takes to compute a job, which means the Grid fills and resources become sufficiently scarce as per a time sensitive, high utilization scenario presented earlier. Each provider uses the round robin scheduling algorithm in section 4.1 and a rescheduling based variation in section 4.2.

The virtual machines ran Ubuntu 11.10 (64bit) server, with full virtualization and ran upon 4 physical hosts. The virtual environment was constructed using OpenNebula 2.0 [28] and Xen 4.0.1 [32]. Each head node had 1GB of RAM allocated and worker nodes 768MB. Each processor ran at a speed of 2.4GHz.

The ISQoS Grid uses WS-Agreement for Java v1.0 for the Broker and Provider agreement process. Ganglia 3.2.0 [13] was used as the information provider.

Jobs were setup to be none data intensive and the stage in/out size was only 1 Megabyte. This mitigates issues with considering the network configuration of the virtual cluster on the cloud testbed. The compute size was given as a value of 3,000. This value derives from a reference processor of 3,000 MHz multiplied by an expected duration of 1 minute. This means upon the resources available the tasks are also expected to last approximately 1 minute.

This means that if a job was allocated to a single machine it would take 8 minutes to complete. The due date was hence set no lower than the submission

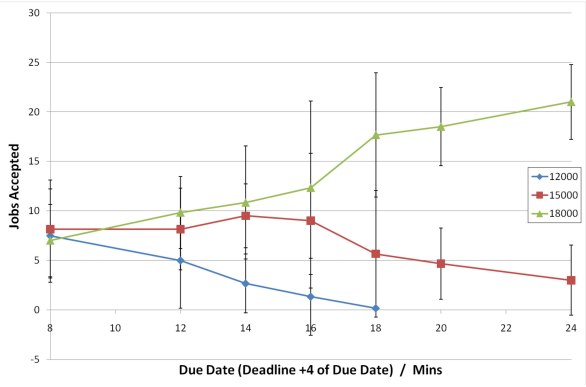

Fig. 3: Transition to Budget Prioritisation - With Round Robin Scheduling

time + 8, with the knowledge that the Grid would soon be overtaxed. The deadline was set to the due date + 4 minutes. We performed 6 runs of each trace and 95% confidence intervals are marked on the graphs. The first 15 accepted jobs of the traces have been ignored to counteract effects of starting with an unloaded Grid.

We establish three different budgets that jobs could be given: 12,000 15,000 and 18,000. These values were chosen as they intersected the likely prices that would be generated at different stages of the experiment. A fixed mark-up of 20% was chosen. A dynamic resource pricing mechanism was chosen which bills time for both the use of network time and resource time equally. It derived its charge from the count of actions that the provider currently has in its schedule, thus tracking current demand. This was chosen as the jobs being submitted were of equal size and there was no great need for a more complex solution. The scheduling algorithm in use was not price aware so a single price could be set for all resources upon a given provider.

4 Experimental Results

4.1 Transition to Economic Constraints Dominance

Figure 3 shows the gradual increasing of the due date and deadline. Initially there is no preference based upon budget shown and the temporal constraints take precedence. When the due date is at 12 minutes the lowest budget jobs at 12,000 start to be penalised and by 16 minutes the lowest budget jobs are all but completely rejected. This is because as the due date increases a greater amount of the work is allowed to be queued concurrently on each provider, causing the resource costs to increase to match the demand.

The jobs with the highest budgets are hence accepted more readily in an ever increasing fashion, whilst the middle budget range jobs increases temporarily as the lowest budget jobs are no longer accepted. This clear prioritisation of jobs

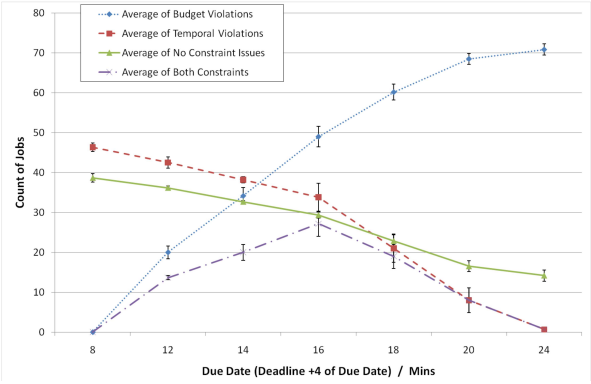

Fig. 4: All Constraint Violations - With Round Robin Scheduling

based upon the budget is seen to be a valuable property of the submission system. This is achieved without any need for jobs to directly compete in an auction style for resources.

In Figure 4, 5 and 6 constraint violations are counted. If the budget causes constraining then a budget violation counter is incremented. A temporal constraint violation is considered to have occurred if the time before the breakeven point is less than zero i.e. where the job is no longer making a profit. This point where the broker breaks even is 16.67% of the way between the due date and deadline [19]. It was useful to pick this point as the deadline was unlikely to be ever passed due to the job admission policy in use filtering out unprofitable jobs. The count of jobs is summed across all runs of the experiment.

In Figure 4 we see the constraint violations are initially of the temporal type only, this is because the highly restrictive temporal constraints are dominating, ensuring queue lengths do not increase sufficiently, which constrains the service prices. As the due date gets larger the budget constraints become more dominant and service price rises.

In Figure 5 and 6 we show the budget pressures for jobs with a set value. The biasing towards the higher priority/budget jobs is hence demonstrated.

Figure 5 shows that initially jobs are either not meeting the temporal constraints or they are accepted. As the due date is increased the budget constraints become dominant. Though temporal constraints are still being violated, it is also the case that the budget constraints are being violated as well, so practically the budget constraints are taking precedence.

Figure 6 like Figure 5 again shows that initially jobs are either not meeting the temporal constraints or they are accepted. As the due date is increased the budget constraints again become more dominant, but it takes much longer for jobs to be predominantly rejected because of budget violations, which is an indication of the desired property of prioritisation.

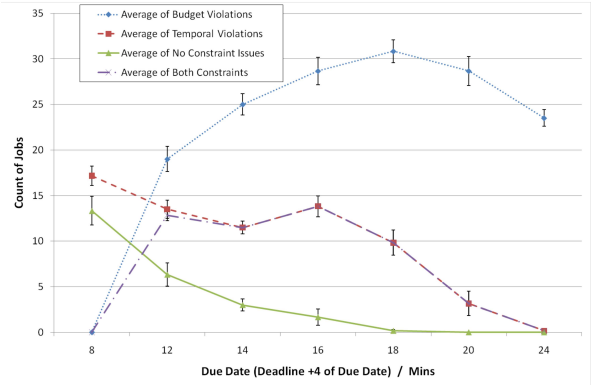

Fig. 5: Constrain Violations with a budget of 12,000 only - With Round Robin Scheduling

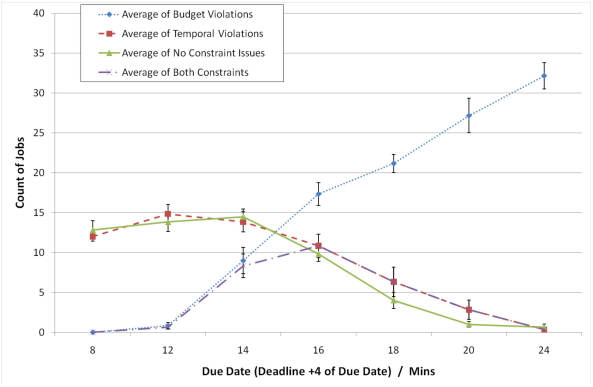

Fig. 6: Constrain Violations with a budget of 15,000 only - With Round Robin Scheduling

4.2 Introducing Rescheduling

We previously indicated in section 2 that different scheduling algorithms may be selected at provider level. This can introduce more complex situations where rescheduling may be performed. In this case we show how this can have a profound effect upon the market.

In the case where rescheduling is performed similar results can be obtained to the none rescheduling case. There are however notable differences.

Firstly we see in Figure 7 that the budget constraint becomes more dominant than the temporal constraints much later on as in comparison to Figure 4. This is reflected in the distinction between the amount of jobs accepted of each type, where the lower budget jobs are no longer rejected entirely. This is in part caused by fewer jobs being accepted, leading to a slightly lower resource cost. On average

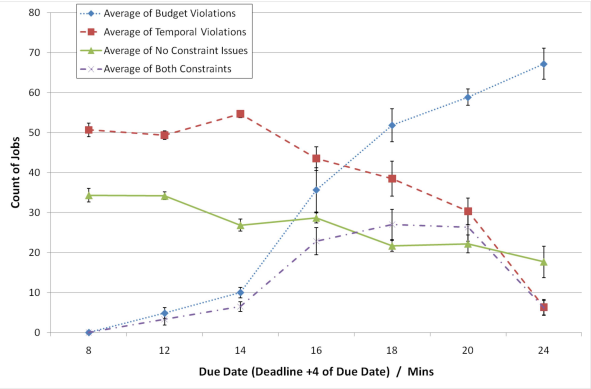

Fig. 7: All Constraint Violations - With Rescheduling

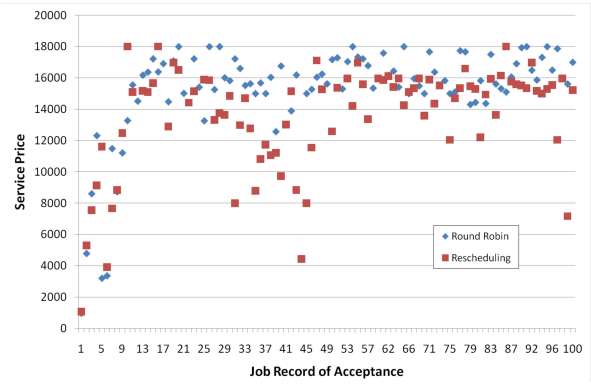

Fig. 8: Effect upon service price of rescheduling (Due Date = 20, Average of all Runs)

28/85 jobs without rescheduling and 26/85 with rescheduling, but it is also more significantly down to greater fluctuations in the service price.

We can see from Figure 8 that the service price when rescheduling occurs drops significantly at various stages of the trace. This means the price is sufficiently low enough for the lower budget jobs to be accepted even when the temporal constraints are at their most relaxed and higher workloads on the server might be expected.

4.3 Price Stability and Selection

In sections 4.1 and 4.2 we saw that before the budget constraints became dominant that selection pressures did not favour jobs based upon the budget available. Hence should a Grid become overworked then in order to make selection

preferences the economic factors should become dominant, by changing the resource/service price. We also saw in section 4.2 how price stability affected the ability of the mechanism to maintain this selection pressure. We therefore suggest various ways that might allow for a more stable price.

The mechanism by which the price is selected may be changed to make an incremental step change from the previous price. This would hence avoid some of the fluctuations and allow for rescheduling. The moderated changes however would have to reflect the completion of spikes in load. Spikes in load can occur for example just before an important conference [16]. The price smoothing mechanism would be required to remain responsive and should not for example artificially maintain a high price at the end of a peak in demand. This is because the system as a whole could either have unrealized profit or utility [25, 21].

The simplest solution would be to ensure that pricing does not rely upon already completed work for pricing. This means any billing event will hence not effect the price, by removing jobs from the queue that are used as part of the measure of current load. An example of such a measure would be to take the difference between the current time and the average completion time of all jobs for the provider.

Alternatively billing events would have to purposefully be held apart. In holding these events further apart it would reduce the number of occurrences, where a lot of work is removed from the queue at the same time. This would require a scheduling algorithm that was geared specifically towards the pricing mechanism. If individual tasks would not be allowed to interweave, then this would help prevent the near simultaneous execution of billing actions. However, if a single very large job completes then regardless of dispersion of billing events the price would fall and potentially harm the selection process.

5 Related Work

Economically oriented Grids have had a long history, including in the early years work such as Nimrod/G [1] and G-Commerce Wolski2001. Nimrod/G is similar in that both our work and theirs intend to serve parameter sweep/bag of task applications, however we have a much stronger focus upon QoS provision.

In more recent years work such upon the SORMA project has created the Economically Enhanced Resource Manager (EERM) [24, 26]. This includes various features such as: demand forecasting, dynamic pricing, SLA formation and the ability to selectively violate less interesting SLAs [24]. We like EERM use dynamic pricing and SLAs but our SLA focuses upon delivery time of the computed results and cost while encoding the acceptable delay into the agreement, negating some of the need for renegotiation. We also use a negotiation mechanism to advise end users of the current state of the Grid instead of rejecting work that has already been accepted.

To cover this subject further a good catalogue of market mechanisms may be found in [25, 7, 4]. A discussion of SLAs and pricing functions may also be found in [33] that covers the subject well. WS-Agreement has previously been used

for job submission in cases such as AssessGrid [11], VIOLA's MetaScheduling Service (MSS) [22] and others [30] [1].

The brokering mechanism we present revolves around its pricing mechanism so it is appropriate to discuss the related pricing models. Early models focus purely on slowdown such as First Reward & Risk Reward [17] and First Price [10], thus are very system centric. Irwin et al indicate a gradient based approach to deadlines to be open to heuristic methods as opposed to hard deadlines [17], which is common to all related models, but is worth mentioning.

Another pitfall we have avoided is that penalty bounds are also not always set, such as in [17, 10] and LibraSLA [9]. This has issues, as pricing mechanisms should have properties such as budget balance and individual rationality among others [31].

First Profit, First Opportunity & First Opportunity Rate [29] like our work uses the same scheduling algorithm to schedule as they do for admission control. However, our broker's mark-up, gives it rationale for participation in the market while also generating a marked difference in providing a boundary of acceptable QoS. This boundary presents itself in both temporal and budget parameters of the model.

The Aggregate Utility [2] model has a lot of flexibility in specifying user requirements but this is at the expense of complexity for the end user. Finally Resource Aware Policy Administrator (RAPA) [14], focuses upon divisible load but has issues in that it caps the maximum deadline in order to limit the maximum penalty paid.

6 Conclusion and Future Work

In conclusion we have discussed the architecture of our broker that establishes a tender market. We have focussed upon describing the submission system and the SLA structure, while also showing the effects of changing the scheduling algorithm in use. We have focussed upon job prioritisation and demonstrated how economic constraints can establish this, whilst also showing an initial period of the temporal constraints holding dominance. Given that users will be aiming to set the budget assigned to jobs as low as possible the budget constraints are likely to always have the desired effect. We finally introduced rescheduling and showed how it can lead to price instability by causing the schedule to have several jobs cleared from it due to completion within close succession. Our future work will investigate how to ensure the stability of the resource price, given the possibility of rescheduling.

[1] A list of WS-Agreement based implementations can be seen at: https://forge.gridforum.org/sf/wiki/do/viewPage/projects.graap-wg/wiki/Implementations

References

1. Abramson, D., Giddy, J., Kotler, L.: High performance parametric modeling with nimrod/g: Killer application for the global grid? In: International Parallel and Distributed Processing Symposium (IPDPS). pp. 520–528. Cancun, Mexico (2000)
2. AuYoung, A., Grit, L., Wiener, J., Wilkes, J.: Service contracts and aggregate utility functions. In: 15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15). IEEE, New York (2005)
3. Battre, D., Hovestadt, M., Kao, O., Keller, A., Voss, K.: Planning-based scheduling for sla-awareness and grid integration. In: Bartk, R. (ed.) PlanSIG 2007 The 26th workshop of the UK Planning and Scheduling Special Interest Group. vol. 1, p. 8. Prague, Czech Republic (2007)
4. Broberg, J., Venugopal, S., Buyya, R.: Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing 6(3), 255–276 (2008)
5. Burke, S., Andreozzi, S., Field, L.: Experiences with the glue information schema in the lcg/egee production grid. Journal of Physics: Conference Series 119(6, 062019) (2008)
6. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proceedings of the IEEE 93(3), 698–714 (2005)
7. Buyya, R.: Economic-based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. thesis, Monash University, Melbourne, Australia (2002)
8. Buyya, R., Giddy, J., Abramson, D.: An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications. In: The Second Workshop on Active Middleware Services (AMS 2000), In conjunction with Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC 2000). Kluwer Academic Press, Pittsburgh, USA (2000)
9. Chee Shin, Y., Buyya, R.: Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In: Cluster Computing, 2005. IEEE International. pp. 1–10 (2005)
10. Chun, B.N., Culler, D.E.: User-centric performance analysis of market-based cluster batch schedulers. In: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on. pp. 30–30 (2002)
11. Djemame, K., Padgett, J., Gourlay, I., Armstrong, D.: Brokering of risk-aware service level agreements in grids. Concurrency and Computation: Practice and Experience 23(13), 1558–1582 (2011)
12. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. Journal of Computer Science and Technology 21(4), 513–520 (2006)
13. Ganglia Project: Ganglia monitoring system (2012), `http://ganglia.sourceforge.net/`
14. Han, Y., Youn, C.H.: A new grid resource management mechanism with resource-aware policy administrator for sla-constrained applications. Future Generation Computer Systems 25(7), 768–778 (2009)
15. Hudert, S., Ludwig, H., Wirtz, G.: Negotiating slas-an approach for a generic negotiation framework for ws-agreement. Journal of Grid Computing 7(2), 225–246 (2009)
16. Iosup, A., Epema, D.: Grid computing workloads. Internet Computing, IEEE 15(2), 19–26 (2011)

17. Irwin, D.E., Grit, L.E., Chase, J.S.: Balancing risk and reward in a market-based task service. In: High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on. pp. 160–169 (2004)
18. Jackson, D., Snell, Q., Clement, M.: Core algorithms of the maui scheduler. In: Job Scheduling Strategies for Parallel Processing, pp. 87–102 (2001)
19. Kavanagh, R., Djemame, K.: A grid broker pricing mechanism for temporal and budget guarantees. In: Thomas, N. (ed.) 8th European Performance Engineering Workshop (EPEW'2011). vol. 6977. Springer, Borrowdale, The Lake District, UK (2011)
20. Kokkinos, P., Varvarigos, E.A.: A framework for providing hard delay guarantees and user fairness in grid computing. Future Generation Computer Systems 25(6), 674–686 (2009)
21. Lai, K.: Markets are dead, long live markets. SIGecom Exch. 5(4), 1–10 (2005)
22. Lehner, W., Meyer, N., Streit, A., Stewart, C., Gruber, R., Keller, V., Thimard, M., Waeldrich, O., Wieder, P., Ziegler, W., Manneback, P.: Integration of grid cost model into iss/viola meta-scheduler environment. In: Euro-Par 2006: Parallel Processing, vol. 4375, pp. 215–224. Springer Berlin / Heidelberg (2007)
23. Liu, C., Baskiyar, S.: A general distributed scalable grid scheduler for independent tasks. Journal of Parallel and Distributed Computing 69(3), 307–314 (2009)
24. Macias, M., Rana, O., Smith, G., Guitart, J., Torres, J.: Maximizing revenue in grid markets using an economically enhanced resource manager. Concurrency and Computation: Practice and Experience 22(14), 1990–2011 (2010)
25. Neumann, D., Ster, J., Weinhardt, C., Nimis, J.: A framework for commercial grids - economic and technical challenges. Journal of Grid Computing 6(3), 325–347 (2008)
26. Neumann, D., Stoesser, J., Anandasivam, A., Borissov, N.: Sorma building an open grid market for grid resource allocation. In: Grid Economics and Business Models, pp. 194–200 (2007)
27. Open Grid Forum: Job submission description language (jsdl) specification, version 1.0 (2005)
28. OpenNebula Project: Opennebula homepage (2012), `http://opennebula.org/`
29. Popovici, F.I., Wilkes, J.: Profitable services in an uncertain world. In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference. pp. 36–36 (2005)
30. Sakellariou, R., Yarmolenko, V.: On the flexibility of ws-agreement for job submission. In: 3rd International workshop on Middleware for grid computing. ACM, Grenoble, France (2005)
31. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: A multiattribute combinatorial exchange for trading grid resources. In: Proceedings of the 12th Research Symposium on Emerging Electronic Markets (RSEEM) (2005)
32. Systems, C.: Home of the xen hypervisor (2012), `http://www.xen.org/`
33. Wilkes, J.: Utility functions, prices, and negotiation. In: Buyya, R., Bubendorfer, K. (eds.) Market Oriented Grid and Utility Computing, pp. 67–88. No. Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, Inc (2008)
34. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: Analyzing market-based resource allocation strategies for the computational grid. International Journal of High Performance Computing Applications 15(3), 258–281 (2001)