
Proceedings Paper:
Kavanagh, R and Djemame, K (2012) Negotiated economic grid brokering for quality of service. In: Yeo, S-S, Pan, Y, Lee, YS and Chang, HB, (eds.) Computer Science and its Applications. 4 FTRA International Conference on Computer Science and its applications (CSA-12), 22-25 Nov 2012, Jeju, Korea. Lecture Notes in Electrical Engineering, 203 . Springer , 87 - 96. ISBN 978-94-007-5698-4

https://doi.org/10.1007/978-94-007-5699-1_10

Negotiated Economic Grid Brokering for Quality of Service

Richard Kavanagh and Karim Djemame

School of Computing, University of Leeds,
Leeds, LS2 9JT, UK
{r.kavanagh, k.djemame}@leeds.ac.uk

Abstract. We demonstrate a Grid broker's job submission system and its selection process for finding the provider that is most likely to be able to complete work on time and on budget. We compare several traditional site selection mechanisms with an economic and Quality of Service (QoS) oriented approach. We show how a greater profit and QoS can be achieved if jobs are accepted by the most appropriate provider. We particularly focus upon the benefits of a negotiation process for QoS that enables our selection process to occur.

Keywords: Negotiation, Grid Brokering, Quality of Service, Job Admission Control, Provider Selection

1. Introduction

Grids enable the execution of large and complex programs in a distributed fashion. It is however, common that resources are provisioned in a best effort approach only, with no guarantees placed upon service quality. 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 [1, 2].

In real world commercial and time-critical scientific settings guarantees that computation is going to be completed on time are required. It is therefore important to establish at submission time the requirements of the users in terms of completion time and cost/priority of the work. In establishing and handling this Grids can be moved away from the best-effort service which limits their importance, as users' reluctance to pay or contribute resources for late returning of results is mitigated [3].

We present two motivational scenarios that illustrate this need for time guarantees.
The first is a commercial scenario such as animation where frames maybe computed overnight before the animation team arrive, partial completion of the work delays or stops the team from starting the next day's work [4]. The second scenario is in an academic environment where it is common before conferences for Grids to become overloaded [5]. It therefore makes sense to prioritise jobs based upon when the results are required. In order that prioritisation is provided correctly an economic approach is used to ensure users truthfully indicate their priorities [6, 7].

This paper's main contribution is that we report upon our study of QoS provision due to enhanced job admission control, within our newly implemented Grid brokering system. We demonstrate the improvement in QoS by submitting jobs for estimates in our negotiation based system and then selecting the best provider for computation.

The remaining structure of the paper is as follows. In section 2 we describe the pricing model and illustrate how broker profit relates to QoS provision. In section 3 we discuss the provider selection policies under test. We then in section 4 discuss the experimental setup and report upon the results in section 5. In section 6 we discuss the related works and in section 7 we conclude our work and discuss future work.

2. Pricing Model & Negotiation

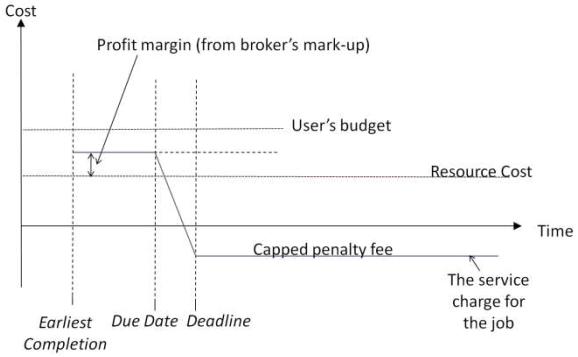

Fig. 1. The Pricing Model

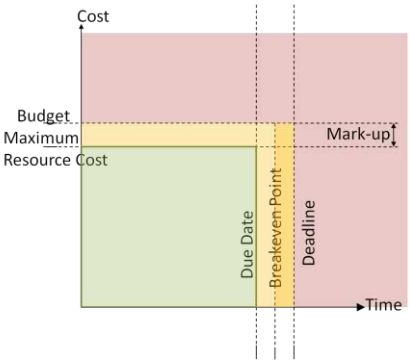

Fig. 2. Offer Evaluation

In this paper we introduce a WS-Agreement (Negotiation) [8] based job submission and brokering system that is part of the ISQoS (Intelligent Scheduling for Quality of Service) broker [9]. In the first stage the broker acquires the job submission templates from each provider. It then fills the templates with the user's preferences. These preferences include:

1. Budget – The user's maximum price they are willing to pay.
2. A due date and deadline – A preferred time and the last point in where the job is still of use.
3. Task description/s – Job Submission Description Language (JSDL) document/s describing the work to be performed.
4. File size and execution requirement – Estimates for each task within a job.

The task descriptions focus upon describing Bag of Task based applications, which are the predominate form of workload upon Grids [5]. We hence use the word job to describe the bag of tasks as a whole. These workloads are formed by sets of tasks that

execute independently of one another, without communication and are hence considered to be "embarrassingly parallel" [10].

The broker then requests offers from providers in the tender market[11]. Each provider calculates a schedule that is suitable for the completion of the work and submits its offer back to the provider. This offer includes the estimated completion time for the job, the overall cost and completion time estimates for each individual task.

The broker then applies a mark-up (see fig 1) performs an assessment and submits the best offer to the user for acceptance. In cases where work is impossible to complete (see fig 2) the broker can send recommendations based upon the existing offers. In this case indicating the increase in time and/or budget that is required in order to complete the work on the Grid under its current state/load. This multiple rounds of negotiation is however out of the scope of this paper and during experimentation we simply reject the job as changing input values to simulate the user's preferences would be highly subjective. It should be noted however providers will not accept work that will go past the deadline so the offer collection phase aids the finding of a suitable provider for the work to be completed upon.

The service charge to the user drops with time after the due-date to a fixed value at the deadline (see fig 1). We chose zero for this cap because it locks the breakeven point to a specific place between the due date and deadline [9]. The service charge is useful as the broker has to pay the providers for the resources used unless a fault occurs or the provider does not perform the required amount of work before the deadline. It also generates a buffer in both economic and temporal terms around the ideal zone for offers, by generating a maximum resource cost before the broker starts using its own markup and a point in time where the job breaks even (see fig 2).

3. Provider Selection Policies

The broker in order to make a profit by generating the appropriate level of QoS must decide which jobs are practical to compute within the allotted time and which provider should compute the job. This brings about various selection strategies for the work to be computed. We introduce several strategies and list them in three categories, namely classical, flooding and selective.

The first classic strategies relate to current mechanisms for submitting to the Grid. They do not require any data from the offers, hence represent a situation with direct submission without negotiation. This can be achieved either randomly or by submitting based upon the current load of the provider.

Randomly: In order to submit randomly offers are first asked for and then an offer is chosen randomly. We chose this way to keep the pattern of submission as similar as

possible to the others in this experiment. The framework does however allow for direct submission thus ignoring the negotiation phase.

Current Load: In this scenario we hook into the Ganglia [12] information provider. We use an average of the cpu_user value across all workers for a given provider. This closely as possible represents if a CPU is busy or not as per the UK's NGS [13] load monitor tool[1]. The user CPU usage is taken so as to ignore as much as possible minor non-Grid system activities taking place upon the worker nodes.

The second set of strategies floods the Grid and tries to optimize greedily upon either time or profit, these represent naïve optimization strategies.

The **Earliest First** and **Highest Profit**: These mechanisms sort the offers (by either profit or completion time) and select the topmost offer. This strategy makes no account for the broker's profit and so long as the budget and the deadline constraints are met then the job is accepted.

The last set of strategies named selective aim to filter out the worst offers and ensure only jobs likely to make the broker sufficient profit are accepted. These mechanisms are **Highest Profit (Profitable Only)**, **Hybrid Offer Filter**, **Load Based Selection (Profitable Only)**, **Random (Profitable Only)** and a **Near Going Rate** mechanism.

Highest Profit (Profitable Only): This extends the highest profit approach and checks to see if the broker will make a profit before accepting.

A **Near Going Rate** mechanism and **Hybrid Offer Filter** [9]: They have been configured to initially sort by profit and select only profitable jobs. The difference from other profit driven strategies is derived from how they perform selection from this sorted list.

Near Going Rate: This establishes from a history of the last n records the current rate at which profit is accumulated. It then establishes a minimum value below this that is acceptable. If the new offer is above this threshold then it is accepted.

Hybrid Offer Filter: If the constraints are fully met then the job is automatically accepted. If the offer is constrained by either time or budget then a going rate assessment is performed. The main aim of this variation is to ensure if the arrival rate slows then unconstrained (fully profitable) job are always accepted. This is particularly advantageous if different mark-ups/priorities are in use and other factors such as differing network transport cost compared to the cost of computation.

Random (Profitable Only) & Load Based Selection (Profitable Only): These extend the classical methods by allowing them to submit to the site chosen by their ranking mechanism and then checking to see if the broker will make a profit.

[1]http://www.ngs.ac.uk/load-monitor
http://nationalgridservice.blogspot.co.uk/2011/03/loaded.html

4. Experimental Setup

We perform experimentation to discover the best selection strategy for selecting between Grid providers, with the aim of enhancing QoS provision. We focus this experimentation upon high load scenarios where correct selection is most required. The high load ensures far more jobs are available than can be computed on time, hence to ensure time constraints are met, which is directly linked to the broker's profit in the pricing model then some jobs must be rejected.

The configuration of the experiments performed is described in this section.

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 being shorter than the time it takes to compute a job meant the Grid fills and resources become scarce as per a time sensitive, high utilization scenario presented earlier. Each provider is configured to use the round robin scheduling algorithm.

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 [14] 2.0 and Xen 4.0.1 [15]. 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 and Ganglia 3.2.0 was used as the information provider.

Jobs were setup to be none data intensive and the *stage in/out size* was 1 megabyte. This mitigates issues with considering the network configuration of the virtual cluster on the cloud testbed. The *compute size* of a job was set to 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, tasks are expected to last approximately 1 minute and that if a job was allocated to a single machine it would take 8 minutes to complete.

Each job's *due date* was set to the submission time + 8 and its *deadline* was set to the submission time + 12, with the knowledge that the Grid would soon be overtaxed.
Each job was given a *budget* of 20,000 which was chosen to be sufficiently high so as not to act as a selection pressure. A fixed *mark-up* for the broker of 20% was chosen, which means the broker breaks even 16.67% of the way between the due date and deadline [9], so the provider must complete work before this point to remain in profit. A static resource price was chosen that bills time for both the use of network and resource time equally at 1 unit per second.

We performed 6 runs of each trace that is used in the experiment 95% confidence intervals are marked on the graphs. The first 9 accepted jobs of the traces have been ignored to counteract effects of starting with an unloaded Grid.

5. Results

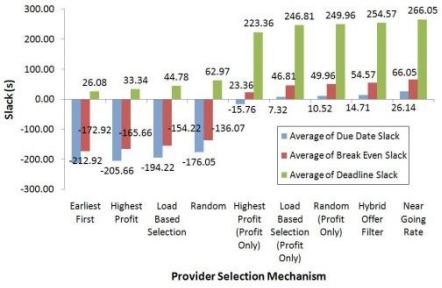

Fig. 3. Average Slack

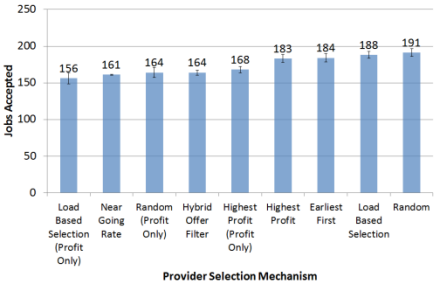

Fig. 4. Job Acceptance

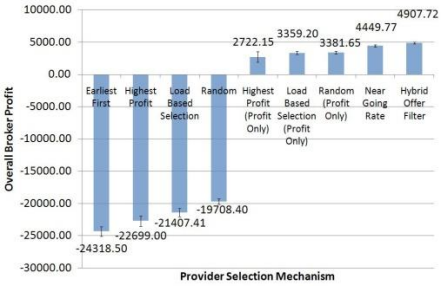

Fig. 5. Overall Broker Profit

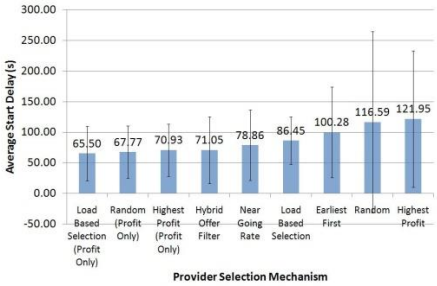

Fig. 6. Average Start Delay

In this section we look at several key metrics aiming at service quality and suitability for the broker, namely the job acceptance, slack, start delay and the overall profit.

The broker's profit directly relates to meeting the QoS requirements, in fig 3 we observe a distinction between mechanisms where profit checking is permissible or not. Highest profit (profitable only) tends to go past the due date making it less suitable. Adaptations of classical submission strategies do well, but tend to have a wide variance in slack and job acceptance (fig 4) as compared to the Hybrid Offer filter. This is also reflected in the overall profit (fig 5), with the Hybrid and Going Rate approaches winning out, some 31.6% above their nearest rivals. The Hybrid approach however works much better than the going rate in lower arrival rate situations [9]. The load based and random selection mechanisms appear to be very similar. It is suspected that the load based selection mechanism does not accurately reflect the queue length/amount of work to be performed when nearing full capacity (as per the experiment), hence acts more as a means of random number generation.

The start delay (fig 6), is used here as a metric for understanding the pressures upon resources on the Grid. Selection based strategies fair best, while random and the highest profitable job strategies perform worst with notable variance. The deviation from the ordering as compared to how many jobs accepted should also be noted as it gives some notion of the differing quality of site selection.

6. Related Work

The brokering mechanism we present revolves around its pricing mechanism so we focus our discussion there. Related models rarely capture the user's real requirements, as we have done. Early models focus purely upon slowdown such as First Reward & Risk Reward [16] and First Price [17], thus are very system centric. Another pitfall we've avoided is that penalty bounds are also not always set, such as in [16, 17] and LibraSLA [18]. Pricing mechanisms however, should have properties such as budget balance and individual rationality among others [19]. First Profit, First Opportunity & First Opportunity Rate [20] like our work uses the same scheduling algorithm to schedule as they do for admission control. However, our broker's mark-up, gives it rational for participation in the market while also generating a marked difference in providing a boundary of acceptable QoS. The Aggregate Utility [4] model has a lot of flexibility in specifying user requirements at the expense of complexity for the end user. Resource Aware Policy Administrator (RAPA) [21], focuses upon divisible load and caps the maximum deadline in order to limit the maximum penalty paid. Nimrod/G [22] is a early work with a limited pricing mechanism and no SLAs.

7. Conclusions and Future Work

We have shown how classical job submission strategies do not fare well in a QoS oriented approach even when providers do not accept jobs past their deadline requirement. Filtering upon profit that is directly linked to QoS vastly improves the situation. The correct use of the pricing model for job selection so that it reflects future scheduled work also significant enhances QoS provision. Our future work will include dynamic pricing to reflect the current Grid workload better, performing tests upon a bigger Grid infrastructure and investigating deployment in the Cloud.

8. References

1. Liu, C. and S. Baskiyar, *A general distributed scalable grid scheduler for independent tasks.* Journal of Parallel and Distributed Computing, 2009. **69**(3): p. 307-314.
2. Battre, D., et al. *Planning-based Scheduling for SLA-awareness and Grid Integration*. in *PlanSIG 2007 The 26th workshop of the UK Planning and Scheduling Special Interest Group*. 2007. Prague, Czech Republic.
3. Kokkinos, P. and E.A. Varvarigos, *A framework for providing hard delay guarantees and user fairness in Grid computing.* Future Generation Computer Systems, 2009. **25**(6): p. 674-686.
4. AuYoung, A., et al., *Service contracts and aggregate utility functions*, in *15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15)*. 2005, IEEE: New York. p. 119-131.

5. Iosup, A. and D. Epema, *Grid Computing Workloads.* Internet Computing, IEEE, 2011. **15**(2): p. 19-26.
6. Buyya, R., D. Abramson, and S. Venugopal, *The Grid economy.* Proceedings of the IEEE, 2005. **93**(3): p. 698-714.
7. Lai, K., *Markets are dead, long live markets.* SIGecom Exch., 2005. **5**(4): p. 1-10.
8. Open Grid Forum, *WS-Agreement Negotiation Version 1.0*. 2011. p. 63.
9. Kavanagh, R. and K. Djemame. *A Grid Broker Pricing Mechanism for Temporal and Budget Guarantees*. in *8th European Performance Engineering Workshop (EPEW'2011)*. 2011. Borrowdale, The Lake District, UK: Springer.
10. Lee, Y.C. and A.Y. Zomaya, *Practical Scheduling of Bag-of-Tasks Applications on Grids with Dynamic Resilience.* Computers, IEEE Transactions on, 2007. **56**(6): p. 815-825.
11. Buyya, R., et al., *Economic models for resource management and scheduling in Grid computing.* Concurrency and Computation: Practice and Experience, 2002. **14**(13-15): p. 1507-1542.
12. Ganglia Project. *Ganglia Monitoring System*. 2012; Available from: http://ganglia.sourceforge.net/.
13. NGS. *National Grid Service*. 2009; Available from: http://www.ngs.ac.uk.
14. OpenNebula Project. *OpenNebula Homepage*. 2012; Available from: http://opennebula.org/.
15. Citrix Systems. *Home of the Xen hypervisor*. 2012; Available from: http://www.xen.org/.
16. Irwin, D.E., L.E. Grit, and J.S. Chase. *Balancing risk and reward in a market-based task service*. in *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*. 2004.
17. Chun, B.N. and D.E. Culler. *User-Centric Performance Analysis of Market-Based Cluster Batch Schedulers*. in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*. 2002.
18. Chee Shin, Y. and R. Buyya. *Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility*. in *Cluster Computing, 2005. IEEE International*. 2005.
19. Schnizler, B., et al., *Trading grid services - a multi-attribute combinatorial approach.* European Journal of Operational Research, 2008. **187**(3): p. 943-961.
20. Popovici, F.I. and J. Wilkes. *Profitable services in an uncertain world*. in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*. 2005.
21. Han, Y. and C.-H. Youn, *A new grid resource management mechanism with resource-aware policy administrator for SLA-constrained applications.* Future Generation Computer Systems, 2009. **25**(7): p. 768-778.
22. Abramson, D., J. Giddy, and L. Kotler. *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?* in *International Parallel and Distributed Processing Symposium (IPDPS)*. 2000. Cancun, Mexico.