UNIVERSITY OF LEEDS

This is a repository copy of *An economic market for the brokering of time and budget guarantees*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/79954/

Version: Accepted Version

Article:
Kavanagh, R and Djemame, K (2014) An economic market for the brokering of time and budget guarantees. Concurrency and Computation: Practice and Experience. ISSN 1532-0634

https://doi.org/10.1002/cpe.3247

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

An Economic Market for the Brokering of Time and Budget Guarantees

July 28, 2014

Abstract

Grids offer best effort services to users. Service level agreements (SLAs) offer the opportunity to provide guarantees upon services offered, in such a way that it captures the users requirements, while also considering concerns of the service providers. This is achieved via a process of converging requirements and service costs values from both sides towards an agreement. This paper presents the ISQoS market oriented mechanism for brokering guarantees upon completion time and cost for jobs submitted to a batch oriented compute service. WS-Agreement (Negotiation) is used along with the planning of schedules to determining pricing, ensuring jobs become prioritised depending on their budget constraints. An evaluation is performed to demonstrate how market mechanisms can be used to achieve this, whilst also showing the effects that scheduling algorithms can have upon the market in terms of rescheduling. The evaluation is completed with a comparison of the broker's capabilities in relation to the literature.

1 Introduction

Advances in Grid computing have in the recent years resulted in considerable commercial interest in utilizing Grid and related infrastructures to support applications and services. However, there still remains the need to correctly market such services and derive quality of service (QoS) provision, as a means to offer compute cycles as a service. This compute as a service as provisioned in Grids is often is often the batch oriented, bag of task applications [1] which still remain of great commercial and scientific interest. Significant developments in the areas of economics for the Grid and defining the available QoS are needed, before widespread adoption outside academia can become a reality of such a job execution service.

In terms of QoS provision it is common within existing Grid infrastructures for applications to be served in a best effort approach only, with no guarantees placed upon the service quality. This is primarily due to the system centric nature of existing production Grids. The emphasis being upon the queuing of jobs

ready for computation, with the sole intent of maintaining high resource utilisation, rather than user satisfaction. This therefore acts as a significant problem to providing QoS should the existing infrastructure be commoditised and offered as a service. User-centric behaviour [2, 3] on the other hand is oriented toward providing the greatest utility to the end user by providing guarantees upon performance and are usually aided by a system of rewards and penalties [4].

The research which is based on the Intelligent Scheduling for Quality of Service (ISQoS) project [5] aimed at addressing the issue of the lack of QoS in Grids by incorporating market mechanisms that classify the QoS obtained in terms of task completion time and service price and by provisioning compute services via scheduling jobs instead of queuing. This ensures that the quality of service received is quantified in a way that reflects the use of the results obtained. This gives the advantage that existing Grid infrastructure can be moved away from the best effort service which limits their importance. This work is strongly motivated towards providing time and cost guarantees, which are further illustrated by the following three motivational scenarios. The first is a commercial scenario, such as animation, where frames may be computed overnight before the animation team arrives for the next day's work, like with any overnight batch jobs partial completion of the work delays or stops the team from starting the next day's work [6]. This scenario presents a commercialisation of the Grid, where the use of best effort services, which have no attached guarantees limits the economic importance of Grids. This is due to users reluctance to pay, directly or indirectly (e.g., by contributing resources to the Grid), for the service they receive, if there is no guarantees on performance particularly when computed results are needed for a commercial process [7]. A further problem is that current Grid markets are system centric and selling time in CPU hours. This however, is not relevant to enterprise customers who have deadlines to meet and have limited ideas about how many physical resources are required to meet such deadlines [8], thus a more user centric approach must be realised. The second scenario is in an academic environment where it is common before conferences for Grids to become overloaded [1]. This requires on an academic Grid for jobs associated with the conference to be prioritised and the results obtained in a timely manner before it becomes too late to submit to the conference. The emphasis in this scenario is upon the limited availability of resources and the requirement to prioritise work, to ensure the most valuable jobs are completed first. In existing infrastructure jobs are queued and results are returned as and when they are ready. If the Grid becomes overloaded jobs that are of a high priority get delayed by jobs that could otherwise have waited. In order to determine priority however users need to be incentivised to reduce their priority, over others. An economic approach is a means to achieve this [4], as value-oriented approaches are not sufficient, as all participants have to be willing to report their priorities and values honestly [8]. The third scenario is taken from the field of Urgent Computing. The aim of Urgent Computing is to aid decision making in events [9] e.g. earthquakes prediction [10]. In such cases it is not useful to waste time waiting for jobs to complete in a queue, as is the

case in the best effort approach. This is principally because Urgent Computing jobs are computationally demanding jobs which are started with a low level of predictability and which need to be completed before a given deadline [9]. In this scenario the time criticality is key. If deadlines are missed then the value of the results gained by computation is negligible. Thus further emphasis can be brought upon ensuring time guarantees.

The market mechanism introduced incorporates the use of service level agreements (SLA) in which the the agreement process reflects the current conditions on the Grid. This paper describes the broker and the mechanisms built that facilitates market based assignments of Grid jobs to resources and of the SLAs formation between Grid services, consumers and providers. This not only secures the interests of the users in terms of time and cost guarantees but it also ensures that the users are aware of and respond to the current resources usage demands. Thus both system and user centric concerns can be catered for. It is also believed that this work though in the context of Grid is applicable to any distributed system such as Clouds where compute tasks are submitted as a batch and results are required within a given deadline for a set price. The main contributions of this work are therefore:

- the ISQoS market mechanism and architecture for job submissions via SLA formation to guarantee time and cost constraints.
- to demonstrate in depth how an economic market for compute services works in regards to the time and cost constraints.
- to relate market pressures to system behaviour such as rescheduling.

The remainder of this paper is structured as follows: Section 2 presents the related work. Section 3 explains the vision of an economic broker in Grid computing. Section 4 introduces the economic model, that drives the job selection process. Section 5 presents the system architecture, including the SLA structure and the flow of events within the software components during agreement formation. This is followed by the evaluation in Section 6.

2 Related Work

The approach in this paper focuses on brokering of in an economic context within Grids. The focus of the related works therefore concentrates upon related economic models and brokerage.

The work proposed here deals with time and cost guarantees, of which there are several that takes such an approach. Early models such as First Price [2] considers pricing upon clusters. A maximum price for a job is first set, by using a first-price sealed-bid auction[11], where each bidder submits one bid without knowing the others' bids. The highest bidder then wins and pays the service price indicated in their bid upon successful job completion. The service price is then linked to slowdown and decreases depending on what the completion time

would have been if the cluster had been dedicated to the job. This does not however capture the user preference and only expressed the resource cost as a degree of acceptable slowdown relating to non-exclusive access to resources only. It is therefore very system centric and does not express the user's preference for completion of the work. Slowdown also has a strong requirement for a reference system by which a notion of slowdown can be derived, which proves difficult in a Grid as different providers will have different reference systems. It therefore makes the comparison between providers unrealistic.

First Reward [3] and Risk Reward [3] pricing functions like First Price focus upon the minimum runtime and upon slowdown, so holds similar drawbacks. Key differences lie in that they use Vickrey auctions [12] for the winner determination problem, as well as a decay rate for price determination instead. Once a minimum runtime is reached the job's price decays at a set rate, hence it is similar to First Price, though a rate factor is used instead to diminish the service price. The decay in the service price is determined for the i_{th} job as $\text{Price}_i = \text{MaxValue}_i \ (\text{delay}_i \times \text{decay}_i)$. This however remains system centric as a job is classed as on time if it meets its runtime estimate. There is therefore no focus upon a user's preference for completion time and as different providers could have resources of different speeds, inter-provider estimates are unlikely to hold much meaning. The loss that is generated by the decay in the service price is also not required to be bound, creating a situation that allows for unlimited penalties, thus it does not have properties such as budget balance and individual rationality[13, 14]. Budget balance is required from actors within a market as long term deficits must be subsidised making them infeasible. In cases where penalties are without limit budget balance could never be guaranteed and is at severe risk. Individual rationality is required equally as the utility caused by participating in the Grid market has to increase, i.e. money gained or utility otherwise there is no need to participate. Hence in cases where penalties are without restriction it can severely damage this rationality. SLAs can also be perceived as contracts and unlimited liability is impractical in a commercial sense [15]. Risk Reward extends First Reward in that it uses a concept called *present value* as an assessment mechanism for jobs of different durations as longer jobs might delay future more profitable jobs. Present value for the i_{th} job is calculated as: $\text{price}_i \ / \ (1 + (\text{discount\_rate} \times \text{remaining\_processing\_time}_i)$. This can therefore be used to favour smaller jobs, without altering the service price paid by the end user.

In the models First Profit, First Opportunity & First Opportunity Rate [16] the job's service price decays immediately at a set rate until a fixed penalty bound is reached. This decay in starting at submission time hence acts in a similar fashion to slowdown with larger jobs been penalised more than smaller jobs.

There is a distinct tier structure of broker and provider, hence multi-site Grid situations can be realised. Providers offer predictions of the number of resources that are going to be available in the future, their prices and a probability distribution that this resource profile will actually occur in practice to the broker, represented by the tuple <start time, duration, resources, price>. The

sending to the broker so much information seems misguided however. As the broker can take a lesser role and respond to offers for completing work, without understanding how the provider intends to fulfil the offer, thus hiding how the underlying service works.

Admission control is performed by testing if the broker's profit is likely to decrease, if this is the case then the job is rejected otherwise it is accepted. After acceptance jobs are required to be completed regardless of cost, unlike others models that permit a cancellation fee[6]. This therefore risks the broker expending large amounts of resources, rather than potentially cancelling a job.

Similarly to First Reward & Risk Reward, LibraSLA[17] has the concept of a penalty rate, though the deadline remains distinct from the runtime. This ensures it does not require a reference machine to determine slowdown. LibraSLA has two types of deadline hard and soft. In hard deadlines the job is stopped once the deadline is reached, while in the soft deadline case the penalty slope is used to define compensation for delays in the completion of the work. The highest acceptable delay and hence penalty is unlimited, so it risks system viability in regards to budget balance and individual rationality [13, 14]. A budget is established that indicates the maximum the user is willing to pay and is required to do so should the job complete on time. The service price is calculated in the same way as FirstPrice and FirstReward as budget and max price are equivalent. The reasoning however for the budget always been consumed fully is however not rational as it should the demand that determines price and the user is likely to overestimate the price of any given job. As the budget does not differ from the service price before a soft deadline/due date, LibraSLA can be considered to lacks the economic competition that is normally associated with market based approaches. This is because payment is not linked to resource availability and informed competition between users. The budget/willingness to pay can conceivably be adjusted by the end user dependent upon market conditions, though it lacks clear guidance on determining such market conditions.

Aggregate Utility [6] like LibraSLA considers a deadline and explicitly considers the notion of start-delay tolerance. This is advantageous in that users want jobs to complete by a given time which is not necessarily equal to the system centric notion of the job's runtime. The service price is again set as a gradient that is determined by a set rate of decay, which does not seem user friendly. The penalty unlike others has a maximum bound that also serves as a cancellation penalty after the service provider has accepted a given job.

The pricing function is on a per task basis and jobs are simply treated as an overarching service contract. An aggregate utility function is then used to assess the benefit of all tasks completing as opposed to a proportion of them and awards either a bonus or penalty accordingly.

The work presented also discusses brokerage, given that economics allows the Grid to provide self-management and quality of service for end users.

Nimrod/G represents early work within this field. It is a hierarchical, decentralized, agent-based scheduler [18] for Grids, aimed at running parametric applications by using economic based scheduling[19], over a Globus based infrastructure [9]. It is however not a general solution for discovering resources and

submitting general applications in a Grid environment, as it uses a restrictive parametric declarative language. It has a limited amount of primitive scheduling algorithms available [20] that are economically oriented. Criticism has also been levelled against Nimrod/G and GRid Architecture for Computational Economy (GRACE) based economics in that it does not support a comprehensive and complex market based economy [9]. Nimrod/G has however achieved some commercial interest, in the forked variation EnFuzion scheduler [19], therefore as an early example of economics it is important, but like all other economic based scheduling in Grids it has not achieved significant commercial take-up [44].

In [21] the GridBus broker [22] and Aneka [23] middleware are used to perform economically oriented negotiations for Bag of Task applications. This is quite advanced in that it can generate alternative offers in cases where the original request for work cannot be fulfilled and also uses a deadline and cost as QoS constraints, though it only has use for a single hard deadline [21]. During the submission stage it asks only one provider at once based upon cost constraints and other job requirements so does not establish a market mechanism by asking each provider to bid for work. This means if the first provider can accept the work then it is accepted even if another provider could have potentially given a better offer, which can be seen to be an oversight.

The Venugopal et al. [21] implementation is also not seen to be user oriented and agreements are heavily focussed upon resources oriented requirements of jobs. They detail the reward for fulfilling an offer and penalty for not, along with very rigid descriptions of reservation windows specifying how many CPUs of what speed, with a start and end time for the reservation. This lacks flexibility and an approach that does not explicitly indicate which reservations are needed and merely states when the user wants the job completing by, in a graduated manner is seen as a better approach to negotiation.

Grid-Enabled Medical Simulation Services (GEMSS) [24] is a service oriented Grid aimed at supporting medical simulations. It uses scheduling and supports use of the Maui [25, 26] and COSY [27] schedulers for this purpose. Web Service Level Agreement (WSLA)[28] is used for SLA formation in GEMSS for both advance reservations of resources and for the agreement between providers and clients.

The economic model driving GEMSS uses a start time and single deadline for jobs, as well as a budget. In having only a single deadline value and no gradient it lacks the graduated economic reason for a provider to complete work on time. Treating the SLA in a similar fashion to a contract [15] the terms are harsh if the provider is just a few seconds late. A job is risky if there is little slack between the estimated time of completion and time allowed to complete the job. The lack of a gradient based approach also prevents heuristic approaches [3]. The need for a earliest permitted start time is one that any model would need to justify, which is elusive in a scheduling situation where goods (such as the computed results) do not perish. GEMSS facilitates swappable resource pricing and can perform pricing based upon either a fixed rate or dynamic load.

GEMSS uses a reverse English auction for establishing the service price

between the provider and client which is used in provider selection. This price is used along with a weighted ranking mechanism for the comparison of the job offers QoS parameters. The QoS parameters in use are a jobs start time, completion time and service price. Though there does not seem a strong reason for comparing jobs using weighted ranking, other than that the economic model is not particularly complex, i.e. not incentivising timeliness in a graduated fashion.

Resource Aware Policy Administrator (RAPA) [29] is a Grid resource manager that aims to create SLA's for job admission and is based upon the Globus toolkit. It is similar in its goals to work presented here i.e. deadline based with economics but has substantial limitations in that it has an inflexible market model. Given the proximity in nature it is discussed in detail below.

Jobs in RAPA regarding deadlines may be one of two types, either hard or soft, soft deadlines means the user can accommodate delay, which is fixed to a maximum of +20% of the user's deadline. This 20% is highly restrictive and does not reflect the users true deadline requirements. It principally derives from ensuring the middleware does not run out of money, as the profit policy used allows for notionally unlimited loss as shown by the following formula [29]:

$$profit = maximum\ reward - a\ penalty\ factor \times (completion\ time - deadline) - resource\ cost$$

It is also limited in that resource usage is allocated by a proportional share mechanism whereas it is useful to have various scheduling mechanisms available.

The budget like in Nimrod/G is specified by the user, and represents the maximum the user is willing to pay. In ISQoS the provider's resource pricing mechanism determines the cost of the resources and the user is billed accordingly up to their budget, thus allowing the price to follow demand. In RAPA and Nimrod/G however, this is simply accepted as the amount the user pays if all goes well, thus requiring some mechanism to ensure the users provide a sensible informed value, such as an auction.

GridEcon [30, 31] aimed to define a model for managing complex market-based service oriented Grids, introducing new stakeholders such as risk brokers, SLA monitors and price modellers [9], which provided value added services such as capacity planning and insurance contracts [30].

In the economic framework it provided features such as Bid and Asks, which is similar to the Condor ClassAd mechanism. Bids in GridEcon representing the resources asked for by the buyer and Asks representing the resources made available by the providers. An important element of matching resources to jobs is ensuring the units of trade are common such as asking for n virtual machines for m hours [30].

GridEcon is flexible in that it provides scope for developing either spot markets, where resources are sold and used almost immediately or futures markets, where future capacity is traded, as part of capacity planning.

It used a top down approach, defining business models around the candidate application and the high level goods to be traded. The lower levels of the Grid

infrastructure had the liability to meet the demands to establish the higher level services that was traded. This approach may suffer from the semantic gap between high level application requirements and low level resource characteristics [9].

AssessGrid Broker [32] is a risk aware broker that implements WS-Agreement, hence there are architectural similarities with the work presented here. The focus of the AssessGrid broker was upon risk and upon assessing of providers, whereas here the SLA formation is different and focuses on job completion time and the economics of job submission. AssessGrid notably had an associated level of risk attached to an SLA and the work that it described. The SLA also included a start and finish time for the work with an associated cost for fulfilling the SLA as well as a penalty associated with failure.

3 The Vision of an Economic Broker

The ISQoS vision is to provide a compute service that can provide Grid services that respect the completion time and cost constraints of end users while also respecting the current resource demands placed upon the Grid.

To achieve this in ISQoS three main actors are utilised. The broker, the provider and the worker nodes. The brokers aim is to query the market and finds the best provider to compute the work on behalf of end users. The provider aims to control local resources/workers so that it can obtain the largest profit for the provisioning of its workers and finally the workers, whose aim is to simply complete the work that is assigned to them.

This tiered model brings about several simplifications in regards to what each distinct element within the ecosystem is aiming to achieve. The broker is the highest tier within the market ecosystem. It is contacted by users as the gateway to the Grid resources. Its principle demands are to find the users the most appropriate resource to compute the task. Its goals are therefore to ensure jobs get completed on *time* and on *budget*. It should therefore concentrate upon market questions and not upon exact details of the job's description and can sit above such implementation details. A broker should offer incentive for its services to be used. It therefore must offer guarantees that are backed up by compensation, thus a broker should also intend to maximise its own *profit*, in so much that it ensures such guarantees are both offered and adhered to.

The provider within the market has the principle aim of planning the work that is to be computed. In a market ecosystem many such providers could be envisaged to exist creating competition within the market. A tender market is chosen as the market mechanism employed, in which the broker asks the provider for tenders on the work to be computed. A providers main goals are therefore to *win work* and to plan its execution. In attempting to win work it must determine how much it is willing to *charge*. This of course is likely to depend upon the nature of the work given and upon its expected *duration*. The provider is therefore concerned with how long each task will be upon each worker and finding the *compatible set* of workers. Aside from computing the du-

ration of reservations upon resources and data transfer times on an appropriate compatible set of resources the provider holds no further interest in the job's description and should confine itself to planning and making itself competitive within the market place. Lastly the worker's only concern is how to execute any tasks it is given, by the provider that organises its workload. It need not therefore consider anything other within the market then the instructions it is given by the provider responsible for it.

4 The Economic Model

The main driver within the ISQoS system is the economic model. It drives the negotiations between the broker and provider's and incentives the provision of results from jobs on time and on budget. It is therefore discussed next, first of all the input parameters, followed by how the service price is calculated as an output parameter of the model and finally the overall effect the model is designed to create.

4.1 Input Parameters

One of the main roles of the economic model is to determine the *service price* the user will pay to a broker for computing jobs. The model will establish a service for computing jobs on time and on budget from underlying resources. Resource providers have a *resource cost* associated with their resources, from which the service price will be based. This will achieve a two tied market, one tier will be a service market for job execution and the second a resource market.

The user will provide a *budget* to a job which is an upper limit to how much can be paid for its completion. They will also provide their time requirements by giving a *due date* and *deadline*. The due date is the date by which the job should complete by and the deadline is the date by which the work would no longer be useful. In doing this a notion of priority can be achieved, for the job in terms of cost and completion time. The notion of a due date and deadline in Grid scheduling is fairly unique, many pricing models use gradients but do not derive this from two unique reference points i.e. a due date and deadline. The benefit of gradient based approaches is that it provides the opportunity for heuristic methods, unlike hard deadlines, which give systems little guidance on how to proceed if no feasible schedule exists for meeting the user's constraints [3].

Given due date, deadline and budget information coupled with *job requirements* it will be possible for a set of schedules to be generated, either by a single provider or by multiple providers that will generate a cost trade off curve that shows the cost to perform a given task in a given time frame. The *job requirements* details the amount of data to be transferred and an estimate by the end user of how long each task of a job will take to compute on a reference processor. A simple calculation of multiplying the time by the clock cycle of the reference processor is used. This assumes that each clock cycle will perform

a similar amount of work. This is known to be simplistic but it is required for the scheduling process and application profiling is out of the scope of this paper. It should also be noted that existing scheduling based Grid systems such as OpenCCS [48, 49, 50] require an estimate of the amount of work to be performed in advance. Given this information a provider can calculate the earliest completion time at this time which indicates the minimum time by which a job may complete (see: figure 1).

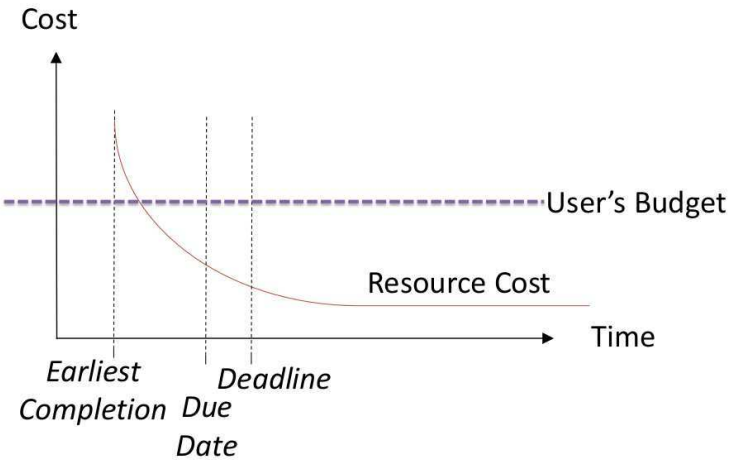

Figure 1: The Resource Cost in Relation to the Pricing Model

A major aim of this model is to provide guaranteed timing for a given price. This therefore means the service price is required to drop if due date is passed i.e. completion time exceeds the due date. This is because it creates an incentive for the broker to select providers that have a schedule that will make the job complete before the due date and deadline.

The broker needs an incentive to participate in the market i.e. individual rationality [14, 13]. It will therefore need to charge an additional amount over the resource cost for its usage. The user will therefore be charged a service price that will be a percentage *mark-up* of the actual cost for resources e.g. +25% of the actual cost of the resources. An alternative charging mechanism would be to have a fixed mark-up from the resource cost. This however, presents an issue in that the lengths of jobs can vary and estimates on task length are also inaccurate [33]. A fixed fee as a percentage of the overall work would be liable to fluctuate. The likelihood of this occurring in Grids means the broker may have reduced incentive to participate in the Grid marketplace.

If the service price drops below the resource cost then there is an additional requirement for an incentive to participate. This requirement arrives from the need to achieve budget balance [14, 13] where the broker must not need any supplementary funds in order for it to continue working. A *cap upon the penalty fee* is therefore introduced in order to control the maximum extent to which the

broker may make a loss upon any single job.

The *mark-up* may either be decided upon by the broker or chosen by the user from a set of choices provided by the broker. The mark-up in affecting the broker's profit can therefore be seen as a mechanism to denote job priority. Mark-up offers the opportunity for tiered pricing where the user can give a priority that is attached to a numeric value of mark-up. This means a value range with low mark-up, a mid range with a medium mark-up and a premium range with high mark-up can be established, in order to prioritise work. This is useful as mark-up affects both profitability and the acceptability of a job. This is because it is a component part of the broker's ability to reschedule a job i.e. if the broker has spare budget and/or the service price is further from the resource cost then it has a greater degree of money available to create such resilience.

In summary the broker presents a service with added value, in that it will guarantee times for a given job or compensate accordingly. The due date and deadline will modify how much of the service charge is paid by the user, in accordance with the mark-up and the budget allocated to the job.

4.2 Calculating the Service Price

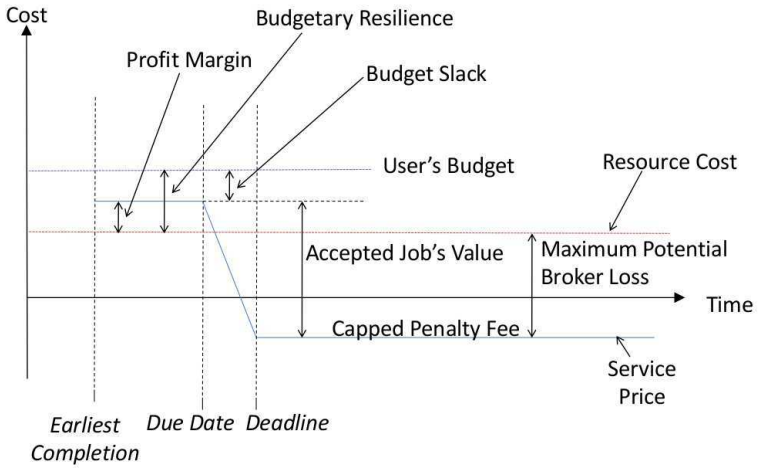

Figure 2: Charging of the Guaranteed Time Service

Figure 2 represents how the price changes in the service market. The service price is based upon the job completion time and is affected by the user's due date and deadline. Once the due date is reached the service price drops linearly until the deadline is reached, at which point any potential loss gets no greater. This is because the deadline is considered to be the point at which the job no longer has any use to the client so the job may be stopped and the service charge/penalty may then be settled.

The service price is decided upon by the broker by selecting an offer that represents a point on the resource cost curve from figure 1. Taking the offer's resource cost it then applies its mark-up. This provides the position of the service price between the earliest completion time and the due date. The cap on the penalty fee then provides the lowest possible point at which the z-curve pricing function may drop to, which is when the deadline is reached.

The resource cost is derived from the offers from the resource market and when the service price and resource cost meet indicates the breakeven point. If the service price is below the resource cost then a loss is incurred by the broker.

The difference between the service price and user's budget provide a measure called *budget slack*, which is important to consider when accepting a schedule.

The difference between the service price and resource cost gives the broker's *profit* margin. It may make sense in certain cases to reduce the amount of profit made in order to ensure the job completes hence avoiding loss. A further value of *budget resilience* can also be given as the difference between the resource cost and the budget, as the broker could notionally sacrifice its profit during rescheduling in order to save the job. This therefore means the profit margin can be seen as the minimum amount money available if the broker is forced to reschedule due to resource failure or slowdown.

An accepted job's worth to the provider may be derived as the difference between the service cost charged and the maximum penalty. This is because liabilities are reduced when an accepted job is completed, (especially if it has a non-zero penalty cap). In order for the broker to minimise its losses the difference between the actual cost of resources and the cap on the penalty fee must be considered. This will be important when rescheduling or deciding upon which jobs should be considered to be dropped from a schedule.

In figure 3 three important measures of how vulnerable a schedule is shown. They may be derived from the difference between the selected schedule's completion time and the following three points:

- due date
- breakeven point
- deadline

These temporal slack measures will hence act as metrics on job offer quality, along with the budgetary resilience and budget slack that was previously discussed, along with more obvious measures such as completion time, service price and broker profit.

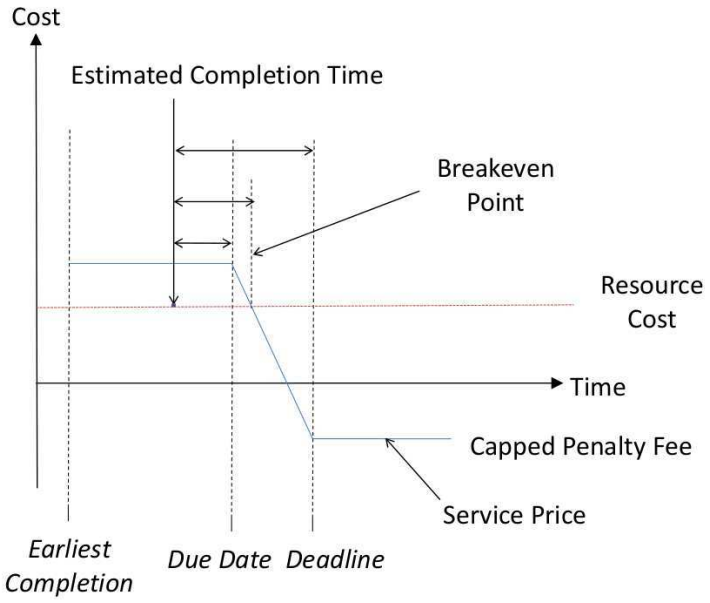

Figure 3: Charging of the Guaranteed Time Service - Temporal Aspects

4.3 Overall Effect of the Market

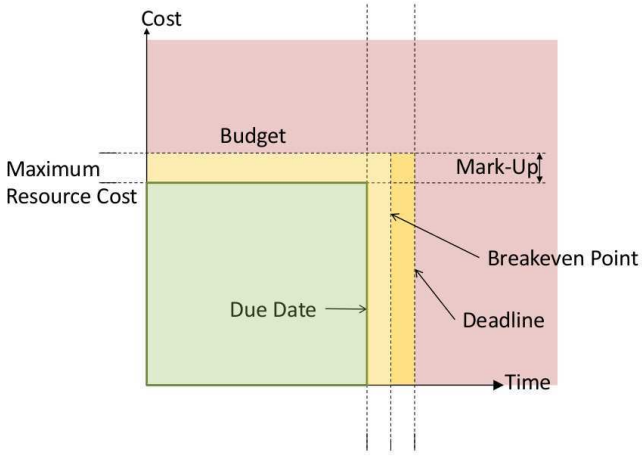

Figure 4: The Offers Market - An Overview of Temporal and Budgetary Aspects

In figure 4 the effect of the temporal and budgetary constraints in the economic model are demonstrated. Firstly the due date and deadline are represented on the x-axis as the two boundaries placed upon completion time. The budget is placed on the y-axis as the upper boundary of the service price.

Given these initial values the mark-up can be used to derive other useful aspects. The maximum resource cost possible where budget slack equals zero can be derived i.e. the broker makes no profit. This leaves the mark-up indicating the minimum amount of budgetary resilience without the broker immediately incurring lost profit. There is also a point in between the due date and deadline where the service price becomes equal to the resource cost and the broker merely breakseven. This therefore completes a border around an offer defining the acceptable service quality.

5 System Architecture

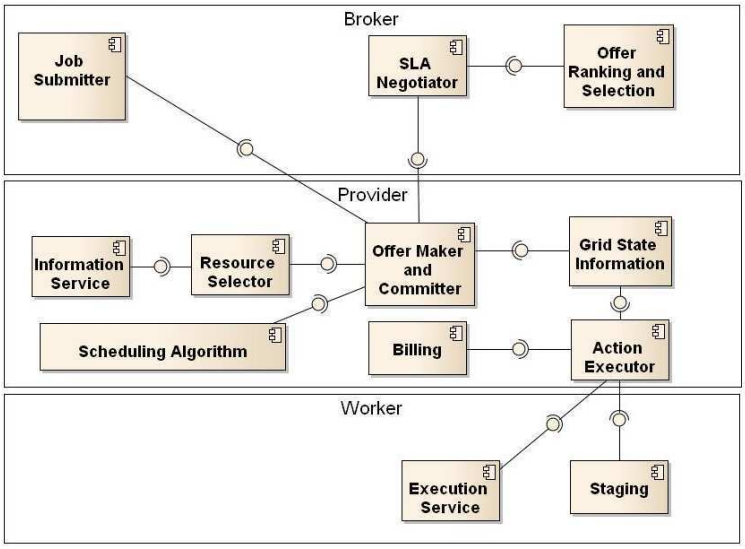

Figure 5: Overview of the ISQoS System

In this section the architecture of the broker is discussed, including in later subsections the agreements structure, the scheduling algorithms in use, offer ranking and selection and finally pricing policies. An overview of the architecture is presented in figure 5. 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. The components shown in figure 5 are utilised in the following fashion.

Jobs are submitted via the job submitter of the broker to the offer maker and committer component of each provider. The broker is expected to contact vari-

ous different resource providers in order to establish a competitive marketplace i.e. multiple providers will compete for work in a tender market.

The information submitted to providers by the broker consists of the job's *resource requirements*, a *budget* that the user has assigned for the completion of a job, the amount of *mark-up* the broker makes for the service, which can be used as a notion of minimum available funds for rescheduling and finally its temporal requirements. These are shown as a *due date* by which the job should be completed by and a *deadline* by when it must be completed. Further details of the agreements structure can be found in section 5.1.

The resource providers from this information perform the initial scheduling of the tasks within a job so they can derive an estimate for completion time and price. A job is considered to be made up of a set of tasks that are all independent of one another. This is because the broker is primarily aimed at parameter sweep / bag of task applications [34], which are the predominant workload upon the Grid [1].

The providers derive their estimates by using a scheduling algorithm (section 5.2) and pricing mechanism (section 5.4) that is plugged into the provider. The price is set by the provider's local pricing mechanism which feeds information into the scheduling algorithm. The scheduler has available to it the current Grid state information and a resource selector component that is capable of selecting the viable set of resources for computing each task within a job.

The schedules that are generated are converted into 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 from the tender market, ranks them and filters out the poor offers i.e. sort by earliest completion time first and filter out unprofitable offers, though various selection mechanisms may be used, which are shown in section 5.3. The broker then takes the "best offer" as determined by ranking and filtering and 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. The winning provider then schedules the work for a final time and then records the new state of the Grid. In the case an offer is not acceptable the user is permitted to resubmit the job with different budget and time requirements in order to acquire a different offer, that may be more acceptable.

Accepted jobs are placed into the schedule which is part of the current state information of the Grid. The state records the mappings between workers and their jobs as well as the current resource and job statuses. 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. The Worker's description parser essentially maintains a set of default questions which can be asked about the job i.e. what is the size of the memory available? or what is the CPU speed? Jobs are also represented by objects that contain XML descriptions taken from the job's associated agreement. The job

description is attached to the agreement as an `xsd:any type`, which allows for XML other than JSDL documents to be used. The descriptions of the tasks are treated in a similar fashion to the resources 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.

A job's description breakdowns down as follows: 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 for the job then the it is submitted for billing.

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

5.1 Agreement Structure

The structure of the agreements that are formed in the ISQoS system are discussed next. The ISQoS Grid architecture is service oriented and is based upon WS-Agreement for Java (WSAG4J) [35]. It is designed in such a way that each provider presents a job execution service. The use of WS-Agreement means requests for work are responded to by providers in the form of offers to complete the work. These offers then can be negotiated upon within a tender contract market. The negotiation and precise structure of these offers is shown in figure 6.

The ISQoS agreement structure like all WS-Agreement based documents has a name assigned to the offer as well as its agreement context. The agreement context details information such as the participants belonging to the agreement. The ISQoS Agreement because it is based upon WS-Negotiation also has a negotiation id and a negotiation context.

The context in terms of the agreement formation process is useful. The providers respond to offers by scheduling, after they have scheduled, they can make an offer. If however the schedule indicates the work cannot meet all the constraints they can set the negotiation context to advisory. This means further work is needed in the negotiation for both sides to agree to execute the job. This allows the ISQoS system to offer advice to the end user about the current state of the market, in terms of price and current workload of the Grid. This of course is expressed in terms of the job the user wants to execute and not a generic statement about a providers current load characteristics.

In the service terms the jobs, budget, due date and deadline requirements are given. There is also a list of tasks in the job that the user wishes to execute. For the purpose of scheduling the task list also contains estimates by the end user of the size of execution that is required and the amount of data that is required for staging in and out. The broker also indicates to the providers what its mark-up is, i.e. its commission. This allows the providers to avoid any schedules that

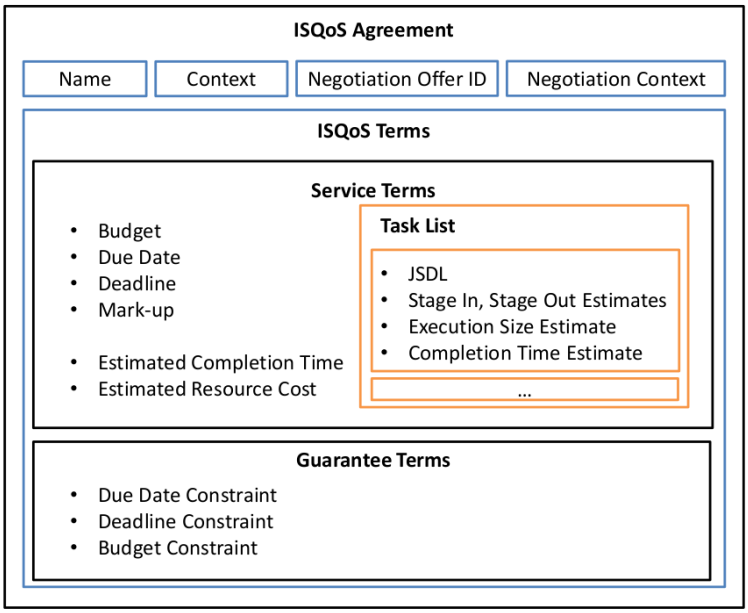

Figure 6: ISQoS SLA Structure

are likely to be rejected and informs providers of the boundaries around the budget and time constraints. These boundaries were previously demonstrated in section 4.3.

The broker focuses upon standards such as JSDL [36]. The agreement document however maintains the capacity to swap out the term language used to describe the work being performed. Thus it need only be 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 [37] or Ganglia [38] based representations of resources, though this will be discussed later.

The final parts of the service terms are for the provider to fill in. Providers are required to give an estimated completion time for each task in the job, along with an overall completion time and service price. These values act as a summary of the job's schedule and does not detail the exact location a job will be executed. The summary values provided here hence can be used by the broker after offers have been returned from the tender market and are the basis of assessments made by the broker, that is shown in section 5.3. The summary values are intended to act in an advisory fashion, so once the provider is accepted by the broker it may schedule the job differently. A provides is however expected to keep to a schedule that is in close proximity to its offer and it should not schedule a job to complete further into the due date to deadline

period than originally offered. This is because the broker will lose money as the price of the execution service gets lower. An intelligent broker that consistently has a job returned later than the offer by an unreliable provider could stop or reduce the amount of work such a provider gets.

5.2 Scheduling Algorithms

The providers use scheduling algorithms to construct schedules, for the purpose of making offers and committing to work. The providers are flexible in which algorithms may be used. In order to allow for a range of experimentation and to show this flexibility a default set of algorithms have been implemented. Algorithms from related work such as Nimrod/G where chosen for this default set, as well as common mechanisms such as round robin and earliest completion time. A Round Robin variation that implements rescheduling is also implemented to assist experimentation that involves rescheduling. The initial available set of algorithms are hence listed as follows:

Nimrod/G's Cost Optimize: [4, 39, 20] This algorithm simply assigns work to the cheapest resource available until it is full i.e. the deadline is reached. This therefore means there is no load levelling which has a significant drawback that if the due date/deadline is sufficiently far in the future then there is no incentive to assign to any resource other than the cheapest. This makes the more expensive resources more likely to be idle and the free space on these resources are simply lost.

Nimrod/G's Time Optimize: [4, 20] This acts like the earliest completion time scheduling algorithm but considers cost, ensuring the budget is not overrun. It achieves this by comparing the cost for completing a task against the available budget per task within the job and only assigns to resources that are less than or equal to the available budget for a given task.

Nimrod/G's Cost Time Optimize: [20, 40] Acts as an improvement to the cost optimize algorithm. It performs grouping based upon cost. It then selects the fastest machine within the cost group. This avoids some of the poor load levelling behaviour of the cost optimize scheduling algorithm, by ensuring it assigns work to all resources with the same price. This however will not cope with costs on a continuous scale and a cost that is very marginally different will be placed into a different grouping.

Cost Per Cycle Optimize: Assigning work to the cheapest resource per cycle is considered best of of such algorithms. This hence acts as a variation upon Nimrod/G's Cost Optimization algorithm. This is considered to be a better way of performing the allocation in that it avoids grouping and falsely identifying the cheapest resources, though it still however lacks load levelling features.

Earliest Completion Time: This creates an earliest time to compute and transfer data matrix and selects the fastest option. It is like Time Optimize algorithm in this regard but omits budget checks. It hence is more suitable for ISQoS as it may be used for offering advice rather than simply failing to provide a schedule.

Round Robin: This performs basic round robin order selection of resources, excluding all resources that are unsuitable for a given task and continuing iteratively until a compatible resource is found. It is used in the experimental section along with the rescheduling variation.

Round Robin Rescheduling: This performs basic round robin order selection and pushes new jobs in as early as possible into the schedule by the moving of existing jobs. The aim of this was to diminish the acceptance preference of jobs based upon how early they arrive. The details of this algorithm are listed in Algorithm 1.

Algorithm 1 Round Robin Rescheduling

```
FOR EACH (Task) {
    Get next worker in round robin order that meets resource
    requirements, skipping the turn of workers that do not meet 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 No action due to start will be moved );
      Place Action in earliest position possible;
      move existing actions later on;
     BREAK;
    }
}
```

5.3 Offer Ranking and Selection

The broker in order to make a profit must generate the appropriate level of QoS. To do this it must decide which jobs are practical to compute within the allotted time and by which provider. This brings about various selection strategies for offers to compute jobs. There are three principle mechanisms by which selection may be performed, these are namely:

- Random: This selection mechanism randomly picks a provider.
- Current load: This mechanism hooks into the information provider and then asks which provider is the least loaded.

- Sort and Filter: This is the most complex mechanism available as it gives a wide variation of options. It first of all sorts offers using a wide variety of possible comparators and then it makes a selection, while filtering out some offers.

The load based and random selection mechanisms also having variations that test for broker profitability, thus ensuring the broker cannot accept work that is expected to cause a loss in profit. The sort and filter selector has some variations of the filtering mechanism that can equally perform a test for profitability. The sort and filter selection mechanism is the most complex out of the available mechanisms due to the possibility of choosing various sorting and filtering mechanisms. The options available are therefore discussed in more detail in sections 5.3.1 and 5.3.2.

5.3.1 Offer Sorting Mechanisms

The sorting mechanisms available are listed below, though this list is implicitly twice the size as the Sort and Filter selector may reverse the order:

- Budgetary Resilience: The difference between the budget and the estimated cost. This hence within the ISQoS pricing model ranks all offers by how much spare money is available to the broker for rescheduling should something go wrong.
- Cheapest: The offers are sorted by their total price.
- Earliest: The offers are sorted by completion time earliest first.
- Profit: The offers are sorted by the broker's profit, most profit to least.
- Profit Rate: This is the broker's profit margin / duration of the job.
- Profit Then Earliest: This first of all sorts by profit and then by completion time.
- Profit Then Latest: This first of all sorts by profit then by completion time in the reverse order.

The comparators that focus on profit and have a second order sort must be used correctly, otherwise the incorrect profit comparators could be selected. This is especially the case given the reversing of the ordering. The Profit then Earliest sort has the following properties:

Natural Order: Profit is least to most and Completion Time is earliest to latest.

Reverse Order: Profit is most to least and Completion Time is latest to earliest.
Caveat: Offers arriving the latest are first.

This is dealt with by the Profit then Latest sort which has the following properties:

Natural Order: Profit is least to most and Completion Time is latest to earliest.

Reverse Order: Profit is most to least and Completion Time is earliest to latest.

5.3.2 Offer Filtering Mechanisms

The offers once sorted need to be filtered and an option selected from the list of available offers. The permitted options for filtering are:

- Minimum Profitability: A minimum acceptable profit is set. The topmost offer that meets this criteria is selected.
- Minimum Profitability Rate: A minimum rate of profit is set for the broker. The topmost offer that meets this criteria is selected.
- Hybrid Offer Filter [41]: as shown in Algorithm 2, is a filtering mechanism that aims to accept all jobs that have no constraint issues and evaluates all others based upon a comparison to a going rate that is established from the last n records of accepted offers. It is hence a hybrid between a going rate filter and a topmost profitable filter. In doing this it has several advantages, which are listed below:

Algorithm 2 ISQoS Hybrid Offer Filter

```
FOR EACH (Offer) {
    Sort the offers based upon the ranking mechanism chosen;
    IF Completion_Time <= Due_Date AND
    Service_Price <= Budget {
      Accept Offer; BREAK;
    } ELSE {
      Take the last n accepted offers and find the average rate
      at which profit accumulates and establish the going rate;
      IF Current_offer_profit_rate >=
      (going_rate − acceptable_deviation_below_going_rate) {
        Accept Offer; BREAK;
      }
    }
}
```

  - If the constraints are fully met then the job is automatically accepted. The main advantage of this is that if the arrival rate of jobs slows then unconstrained (fully profitable) jobs are always accepted. The pricing model dictates that these jobs are the most profitable possible anyway. This can be contrasted with a going rate calculation which may not accept a job with a lower profit even if it could fit with no problem. This is particularly advantageous if different mark-ups are in use and other factors such as differing network transport cost compared to the cost of computation.

  - If offers are constrained by either time or budget then a going rate assessment is performed. If the offer still remains profitable enough then it is accepted, hence if jobs with different mark-up's arrive they will be treated differently and higher mark-up based jobs may still make a sufficient profit to be acceptable.
  - It can be expected on an unused Grid that jobs will be accepted readily, this hence removes the immediate need for a history of records and alleviates the issues of starting with no information from which to determine the going rate.

- Near Going Rate: This establishes from a history of the last n records the current rate at which profit is accumulated by the broker. It then establishes a minimum value below this that is acceptable. If the new offer is above this threshold then it is accepted. The previous jobs are used to establish the going rate. Both the history size and a proximity below the going rate considered to be acceptable are configurable. The advantage of specifying deviation from the going rate over having a minimum profit rate is that it better follows demand and will automatically adapt if the environment changes. An example of this would be if the amount of resources available changed which would allow more space in the Grid, that in turn means more jobs could be accepted causing the service price to change when dynamic pricing is used.

- Topmost Offer: This selects the topmost offer from the sorted list of offers, with no further selection criteria.

- Topmost Profitable Offer: This selects the topmost offer from the sorted list that is profitable. If no offer is acceptable then no option is provided to the end user for execution. The offers would then solely be used to advise the end user how much they could be expected to pay to have work completed on time.

5.4 Pricing Mechanisms

In the ISQoS Grid the providers have to set their own price. This has to be set competitively and it has to reflect the current load the provider is experiencing i.e. price has to follow demand.

This demand is caused by the acceptance of jobs by the provider and is expressed in the provider's schedule. The schedule has a set of jobs which are made up of tasks which are in turn comprised of actions, which have to be assessed for the demand they represent for the provider's resources.

The actions consist of *stage in*, *execute*, *stage out* and *clean*. Stage-out only represents the transferring of files while clean-up cleans and bills the work. The stage in and execute actions may also be combined to form a single action, meaning the worker nodes will automatically start execution once the stage-in has been completed. These actions together establish the demand on the provider and its resources.

This demand the actions create when assessed may be for the provider as a whole or it may be performed so the price is different for each given resource. The provider level price setting mechanisms are therefore:

A count of actions that the provider has to perform: The amount of actions in a provider's schedule is counted and then a price is assigned accordingly. The number of actions is proportional to the number of tasks so it was not seen to be of any use to count tasks. Actions are placed in a queue of actions that the provider is expected to start, so task counting against a list of actions takes more computational effort.

The average finishing time of work on the provider's resources: This would be similar to the "count of provider actions" metric but would account for jobs of varying lengths. The completion time of the last action in the schedule is compared to the current time. This value is then mapped across to a price written in a configuration file on disk.

A static resource price: This is the most basic case where a static singular price is set for the provider as a whole.

This can also be set at a similar fashion for each individual resource/machine:

A count of actions that a particular machine has to perform: A variation upon the "count of provider actions" metric but the count is performed for each resource which is then priced separately. This variation can therefore be used when cost guides local resource selection.

The finishing time of all work on a particular machine: A variation upon the "count of machine actions" metric but would account for jobs of varying lengths. The completion time of the last action in the schedule is compared to the current time and from this value it is then mapped across to a given price.

A static resource price: This is the most basic case where a static singular price is set for the provider as a whole forming a default. Individual named resources are then allowed to take an alternative fixed price.

6 Evaluation

In order to evaluate the proposed model experimentation is presented here that was performed upon a test bed. It concentrates upon job prioritisation and price stability within the ISQoS Grid as a means of demonstrating the market model presented and how it distinguishes between jobs in terms of QoS levels.

6.1 Experimental Details

We first of all describe the experimental method and setup of the resources used during the experimentation and then discuss the settings used in the experimentation.

We established a Grid with 2 providers/clusters of machines. Each provider had 4 virtual machines (VMs) of which one also acted as a head node. Jobs were submitted at intervals, from a separate broker virtual machine instance. This prevented the broker from making one provider less competitive than the other. In total the experiment hence had 9 virtual machines.

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 [42] and Xen 4.0.1 [43]. Each head node had 1GB of RAM allocated and the worker nodes had 768MB and the processors each ran at a speed of 2.4GHz. The head nodes were allowed to be allocated to by the scheduler ensuring the resource space allowed was as large as possible. The size of the Grid was chosen because of the limited resources available. It was also desirable to have competition between providers so 2 providers were used.

The ISQoS Grid which was setup on the VMs uses WS-Agreement for Java v1.0 for the Broker and Provider agreement process. Ganglia 3.2.0 [38] was used to provide resource information to each of the head nodes, about the availability of its resources.

There were 6 runs of each trace taken and 95% confidence intervals are hence marked on the figures shown in the results section. The first 15 accepted jobs of the traces have been ignored to counteract effects of starting with an unloaded Grid.

The experiment was run with 100 jobs with 8 tasks each being sent from the broker. The broker had to select either to accept the job and submit to one of the two providers or reject the job. Jobs were submitted with a 30 second gap between submissions. 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 as discussed in the motivational scenarios.

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 upon the available testbed. The execution size estimate for each task was given as a value of 3,000. This value is provided by the end user. It derives from the user considering a reference processor and estimating how long an application will run for on such a processor. This is known to be simplistic, but profiling applications was seen to be out of scope of this paper. A reference speed of 3,000 MHz is hence multiplied by an expected duration of 1 minute. This means upon the resources on the testbed that the tasks were 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 time + 8, with the knowledge that the Grid would soon be overtaxed. The deadline was set to the due date + 4 minutes.

Three different budgets that jobs could be given were established: 12,000 15,000 and 18,000 and a fixed markup of 20% was chosen. These values were chosen as they intersected the likely prices that would be generated at different stages of the experiment.

During the course of the experimentation rescheduling is introduced. In section 6.2.1 the round robin scheduling algorithm is used. A rescheduling based variation is then used in sections 6.2.2 and 6.2.4. The algorithms implemented by the providers may be found in section 5.2. The round robin scheduling algorithm was chosen because the scope of the paper is brokering and economics and a provider only had to create a schedule in a reliable fashion. A round robin scheduling algorithm also acts in a similar fashion to a queue based system which is in keeping with exiting Grid infrastructures.

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, which maps across to a set price (see section 5.4). The provider thus tracks current demand when setting its price by this method. This method was chosen as the jobs being submitted were of equal size and there was no great need for a more complex solution. If jobs were of a different size it would be required to have a more complex solution as the tasks might not be of comparable size meaning counting actions would not truly reflect current resource demand. The scheduling algorithm in use was not price aware so a single price could be set for all resources upon a given provider. It is possible to count the price either across the provider as a whole or against a single resource. Seen as the algorithms did not account for price i.e. for load levelling purposes there was no need to set prices differently. In section 6.2.4 the pricing mechanism is changed so it ignores work that has been started or finished.

The broker used the hybrid offer filter selection mechanism (see: algorithm 2) and ranked jobs by broker profit, the offer ranking and selection mechanism used is shown in figure 5.3.

6.2 Results and Analysis

6.2.1 Job Prioritisation

A distinct part of any economic model is the ability to drive prioritisation within the market. Figure 7 shows this as a gradual effect as higher budgets become prioritised over lower budgets when the due date and deadline is gradual increased. 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 demonstrates selection based upon budget which is highly preferential and how its emergence can be restricted by the temporal constraints. The emergence of market pressures occurs because as the due date increases a greater amount of the work is allowed to be scheduled concurrently on each provider. This causes the resource costs to increase to match

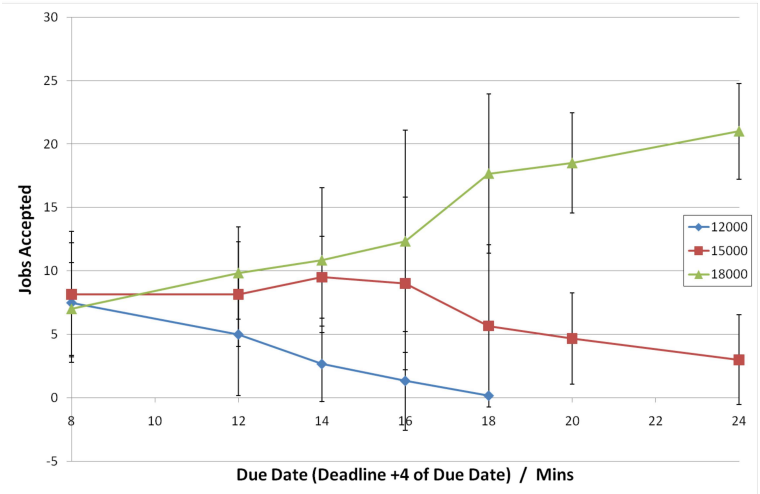

Figure 7: Transition to Budget Prioritisation - With Round Robin Scheduling

the demand. In doing so the lower priority/budget jobs then are priced out of the market.

The jobs with the highest budgets are shown to be 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 prioritisation of jobs based upon the budget is seen to be a valuable property of the submission system especially as it is achieved without any need for jobs to directly compete in an auction style for resources.

In figures 8, 9 and 10 constraint violations are counted. If the budget constrains a job 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 [41]. It was useful to pick this point as the deadline was unlikely to be ever passed due to the hybrid offer filter admission policy in use as it filters out unprofitable jobs, hence any job that would complete too close to the deadline would be rejected.

In figure 8 it can be seen that the constraint violations are initially of the temporal type only, this is because the highly restrictive temporal constraints are dominating, ensuring the overall schedule size does not increase sufficiently, which limits the service price. As the due date gets larger the budget constraints become more dominant as larger schedules cause the service price to rise.

In figure 9 and 10 the selection pressures for jobs with specific budgets, namely 12,000 and 15,000 are shown, demonstrating the biasing towards the higher priority/budget jobs.

Figure 9 shows that initially jobs are either not meeting the temporal con-

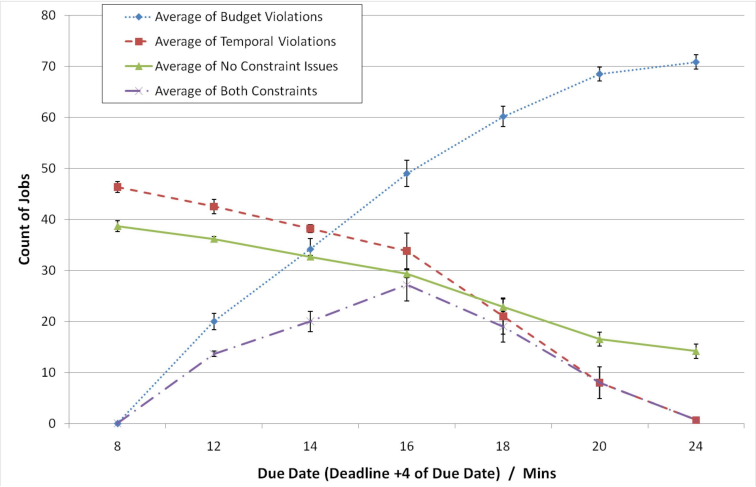

Figure 8: All Constraint Violations - With Round Robin Scheduling

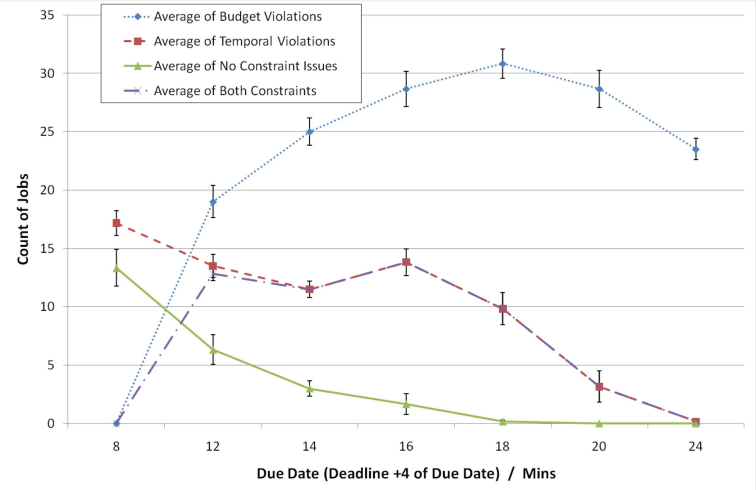

Figure 9: Constraint Violations with a Budget of 12,000 only - With Round Robin Scheduling

straints or they are accepted. As the due date increases 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, this in essence means the budget constraints are taking precedence as violating both constraints is equivalent to only violating one constraint.

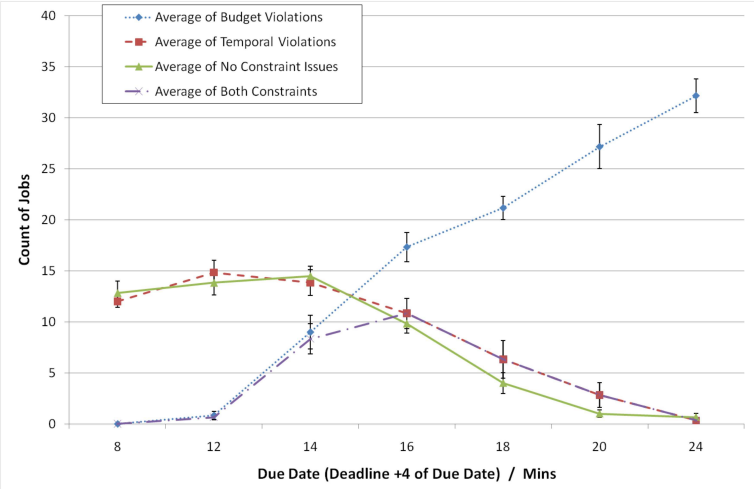

Figure 10: Constraint Violations with a Budget of 15,000 only - With Round Robin Scheduling

Figure 10 like figure 9 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.

Further insight into the effects of increasing budget in terms of selection can be seen in terms of the start delay associated with each billed job is shown in figures 11 and 12. Figure 11 shows values for all due dates and figure 12 for higher due dates of: 18, 20 and 24. It is apparent that the higher budget jobs obtain higher start delays as compared with lower budget jobs. This seems initially counter to what is required, however the start delay shows how long a job has to wait for before it is executed and hence how busy the provider is at the time of billing. Therefore more jobs in the schedule means not only that jobs have to wait longer to be executed, but that the price is higher for a job to be placed in the schedule, due to the higher demand. This means that if a job has a higher budget it has greater scope of being accepted into the larger schedules where it would have otherwise have been rejected. This can be seen when examining figures 11 and 12 side by side. Initially it can be seen in the earlier bills in figure 12 that the lower budget jobs (12,000) are delayed equally as with all other jobs but are then priced out of the market by the time the

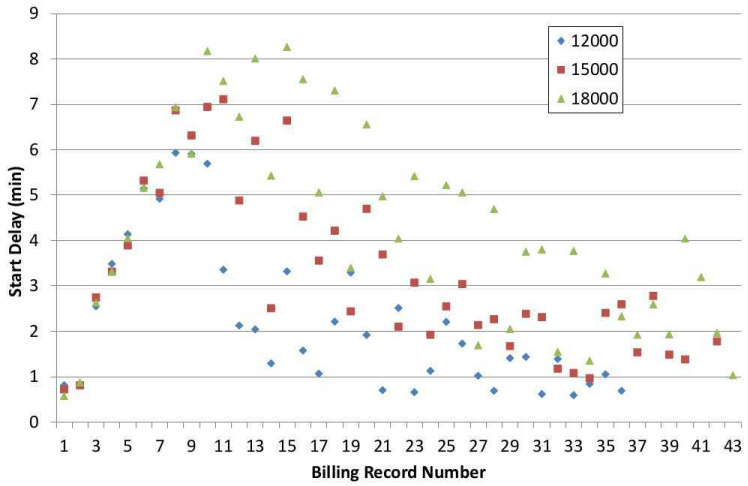

Figure 11: Average Start Delay Across 6 Runs - With Round Robin Scheduling - For All Due Dates

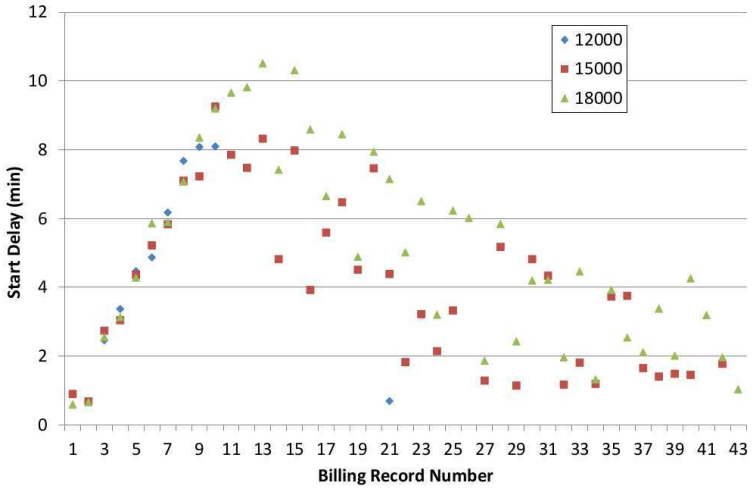

Figure 12: Average Start Delay Across 6 Runs - With Round Robin Scheduling - For Due Dates 18, 20 and 24

Grid reaches full load. It can also be seen that the averages are higher in figure 12 than figure 11 due to the exclusion of the lower due dates, which is discussed next.

Due date also effects the start delay as it simply permits greater delays to be possible as shown in figure 13. The start delay can also be seen to be tending towards a steady state, which is obtained faster when the temporal constraints are made more stringent. This indicates that if production systems were to follow this model that the intent of users to obtain the fastest completion time and setting temporal constraints accordingly is hence likely to cause stabilisation faster in regards to delays incurred.

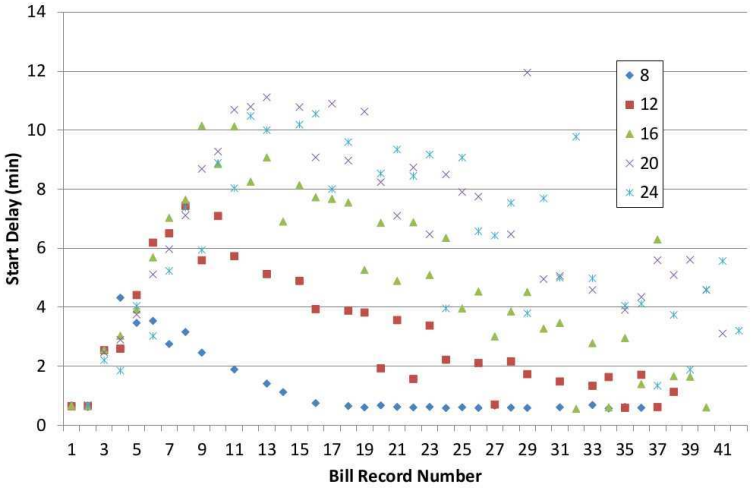

Figure 13: Start Delay - With Round Robin Scheduling - Shown By Due Date

6.2.2 Rescheduling

The ISQoS providers can use different scheduling algorithms as shown in section 5.2. This can introduce more complex situations where rescheduling may be performed. The introduction of scheduling can cause profound effects upon the market, which is shown next. In the case where rescheduling is performed similar results can be obtained to the none rescheduling case. There are however notable differences.

Figure 14 demonstrates that the budget constraint becomes more dominant than the temporal constraints much later on as compared to figure 8. 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, which is considered to be caused by greater fluctuations in the service price.

These fluctuations in service price can be seen in figure 15, which is a trace of the service price over the 6 runs which have a due date of 20. It shows that when rescheduling occurs the service price drops significantly at various stages

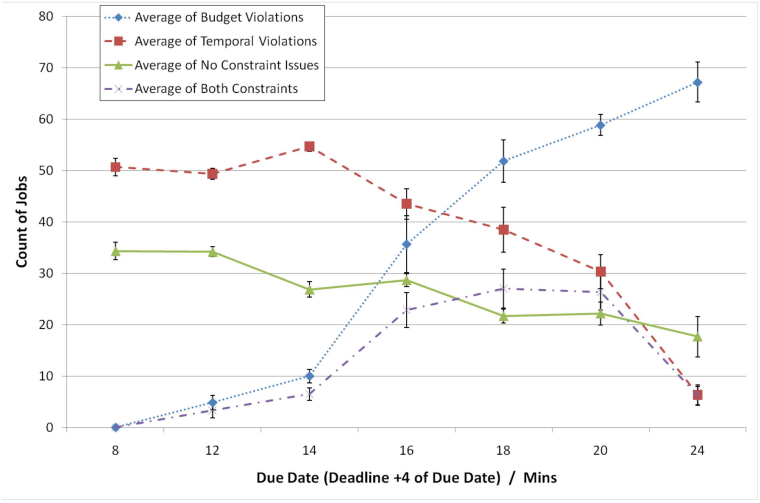

Figure 14: All Constraint Violations - With Rescheduling

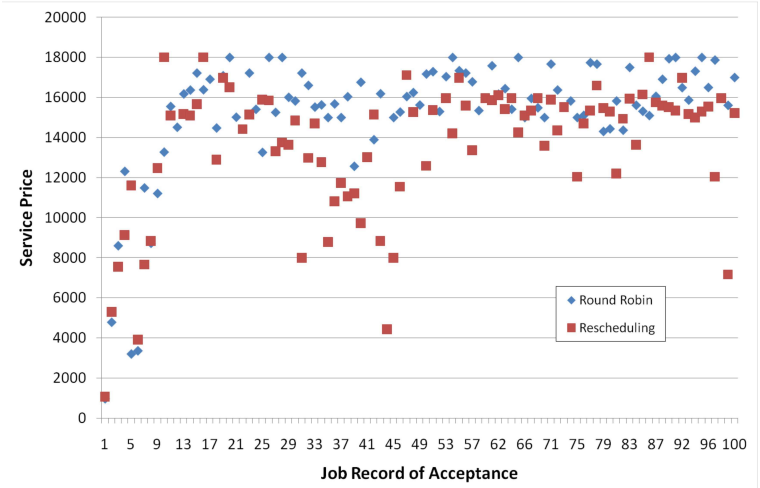

Figure 15: Effect upon Service Price of Rescheduling (Due Date = 20, Average of all Runs)

in the trace, which is not the case where rescheduling is not in use. The drop in price means it becomes low enough for the lower budget jobs to be accepted. This is the case even though the temporal constraints are relaxed meaning that higher workloads on the server could be expected. These higher workloads would therefore give rise to an expectation that the service prices was also high.

6.2.3 Causes of Price Instability

In order offer an explanation for what is happening and why the price is being reduced a simplified case is presented. A single processor is examined, with the arrival of four jobs that submitted in sequence. The temporal constraints is simplified and due date is considered to be equal to the deadline and the time is represented as discrete time rather than continuous. The arrival of each of the four jobs is shown, with the top row representing the processors availability and the rows after representing the jobs availability. The time increments on by one step each time and is represented by the dashed area on the figures presented next. This example is presented for the rescheduling and round robin cases to enable the examination of the differences between the two sample cases.

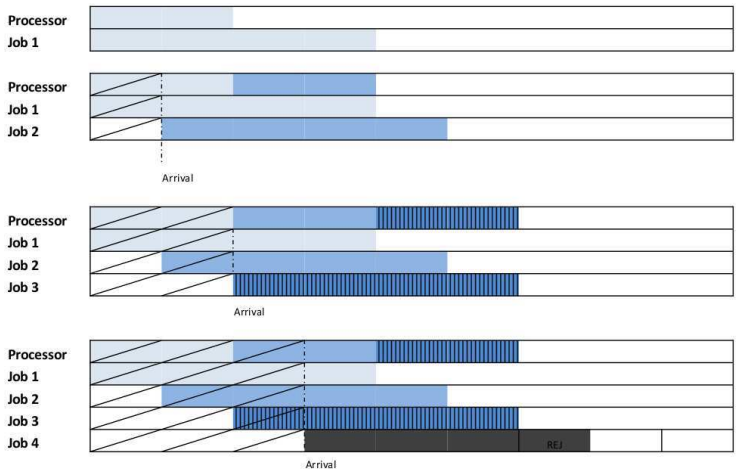

Figure 16: Instability Demonstration - Round Robin

In figure 16 the round robin base case is presented. It can be seen that Job 1 arrives on the processor and takes up two blocks but has a deadline of four ahead of its arrival time. The processor is free so it is allocated immediately. The arrival of Job 2 one block of time later means Job 1 and Job 2's allocable space overlaps. Job 1 having already been allocated takes priority and Job 2 is placed afterwards. Similarly this happens to Job 3. Job 4 arrives and its allocatable

space is all but overlapped by Jobs 2 and 3 and as they arrived first and their reservations are respected, hence Job 4 is rejected. This is unremarkable but is established so the rescheduling case may be examined as well.

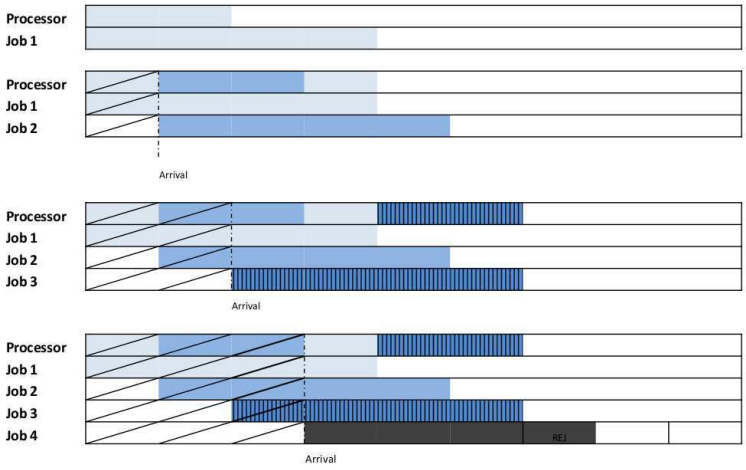

Figure 17: Instability Demonstration - Rescheduling

In figure 17 we see the rescheduling case being demonstrated. The arrival of Job 1 is handled in much the same way as before given the processor is unloaded. The next arrival one block of time later means Job 1 and Job 2's allocable space overlaps. This time in order to break the precedence of earlier jobs always coming first, Job 1 takes a lower priority. The first block/in this case a task has already been executed. The second "block in the job"/task may now however be postponed. Job 2 hence starts immediately in this time period. When Job 3 arrives Job 2 is prevented from moving by the end of Job 1. Job 3 is hence placed at the end of the processors available space. Finally Job 4 arrives and it is again rejected due to a lack of space.

This simple case shows how the jobs that arrived earlier can block future jobs from being moved. It also shows how the billing of each job can be moved closer together. This is important as this removes work from the schedule. Once this has been done it means the service price drops, in cases where all actions in the schedule are used to determine current load. Rescheduling lends itself towards two jobs completing on a single resource within short succession, which rapidly reduces the service price.

6.2.4 Price Stability

Given there is instability in the service price and given the explanation presented in 6.2.3 a mechanism to alleviate the problems is tested. Comparisons are therefore made between the round robin, rescheduling and the re-priced (rescheduling) variations.

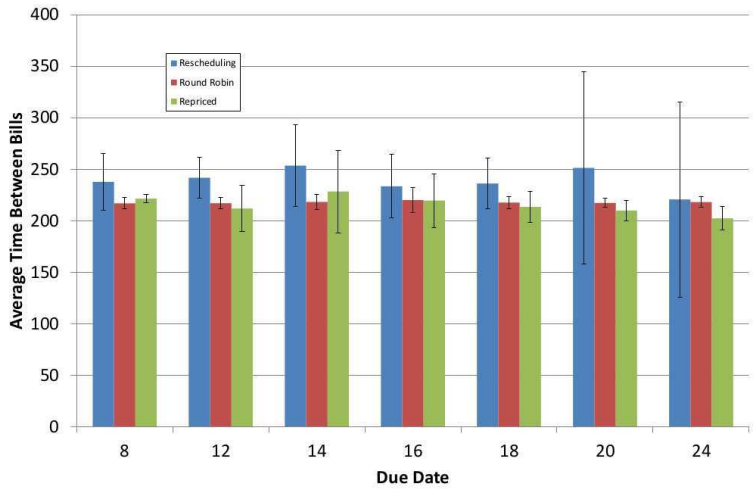

Figure 18: Average Time Between Billing Events for each Given Due Date)

To further promote the explanation given in 6.2.3 the time between billing events is shown in figure 18. It can be seen the time between billing events does not differ that greatly between any of the three series before due dates 20 and 24. It can then be seen that the rescheduling variation gets a much greater range of possible values and the average from one run to the next differs markedly. Essentially the average time between bills varies much more in the rescheduling case then it does in either the round robin or repriced variations.

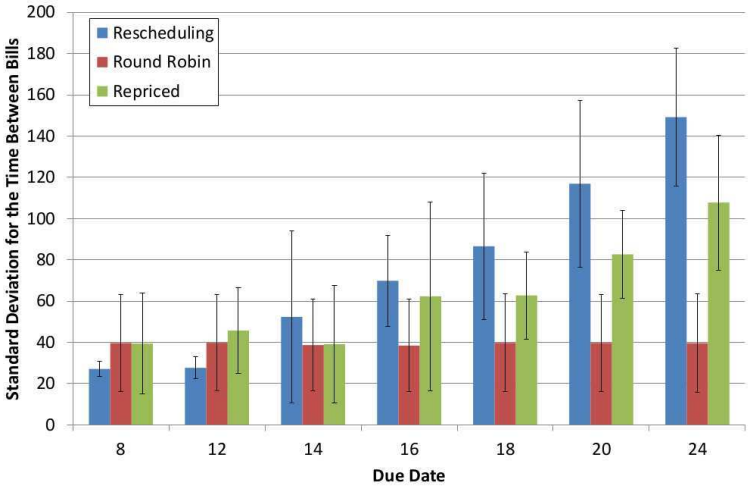

Figure 19: Standard Deviation for the Time between Billing Events for each Given Due Date)

To continue the premise that rescheduling causes billing to removed jobs from the schedule at similar times, thus causing price drops, the standard deviation for the time between bills is shown in figure 19. The standard deviation for both the rescheduling case and the repriced increases but the rescheduling case increases much quicker. In considering 19 and 18, it can be seen that the average time for the Round Robin and Repriced variations does not move, while the rescheduling variation does. It can also be seen that the time between bills in the rescheduling and the repriced variation begins to vary more with larger due dates, though the repriced variation maintains the average better. This maintenance of the average means the price is likely to be more stable.

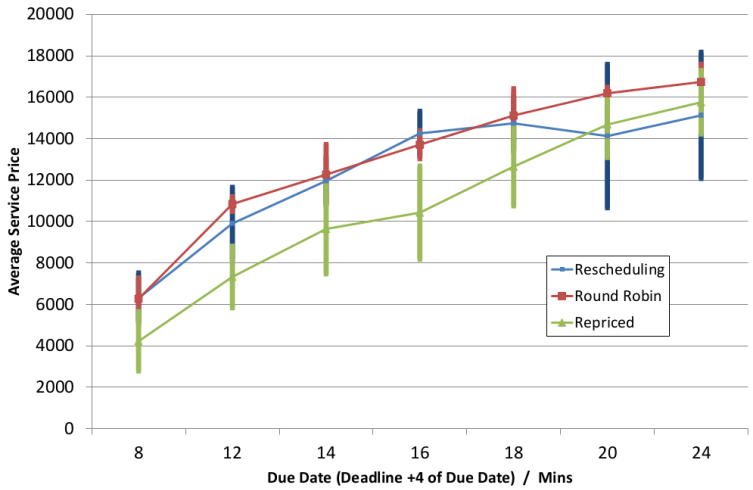

Figure 20: Average Service Price (For Each Due Date)

In testing the instability consideration also has to fall on the average service price over each of the runs (see figure 20). The service price for rescheduling and round robin are about equal/and within error margins until the later due dates of 20 and 24, which mirrors what is happening with the time between bills. It can be deduced that the only thing that distinguishes round robin and the rescheduling variation is related to how the rescheduling occurs i.e. something happens due to rescheduling, or the ineffectiveness of the rescheduling algorithm. The rescheduling variation cannot therefore be efficiently performing the rescheduling with respect to the pricing model, causing the service price to go down.

It can also be seen in figure 20 that the re-priced rescheduling variation does not initially perform as well in terms of service price. This is not surprising as actions that have been executed do not contribute to the service price, hence it is at a disadvantage. It can be observed that as the due date increases, this effect is diminished. This is due to larger schedules, which ensure the work that has been executed and is yet to be billed makes up a smaller proportion of the schedule.

This disadvantage should however exist throughout and it could be expected that its average service price should never approach that of the rescheduling variation. It however surpasses the service price of the plain rescheduling variation, which should always be in a better position to obtain a higher service price.

It should also be noted that there is a possibility as the maximum budget possible is 18,000 that the service price is tending towards a maximum in an asymptotic fashion i.e. it will not reach this maximum value, but will approach very close. The re-priced variation could therefore be less susceptible during its approach to this maximum.

Hence it can be seen that it performed very similarly to re-priced and round robin until this point. We therefore introduce figure 21 which examines the service price over time in the same fashion as figure 15.

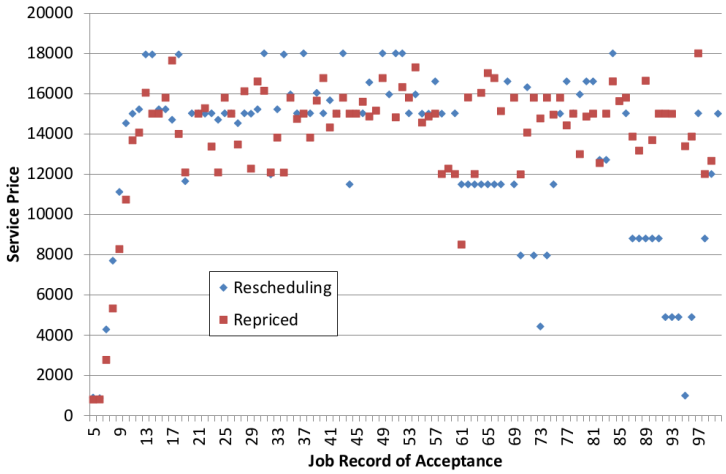

Figure 21: Service Price over Time (Due Date = 24)

The re-priced variation obtains a similar service price to the rescheduling variant however more importantly, it does not suffer from random dips in the service price and only one outlier moves away from the 12,000 to 18,000 band of service prices.

6.3 Discussion upon Price Stability

In this section we discuss the stability of prices in relation to dynamic pricing models. Dynamic pricing is the adaptation of the price of resources in accordance with demand. This gives rise to a situation shown in figure 22 where the following is the case:

- The price is set by observing the current schedule for demand information
- The schedule has work added to it which is effected by the price of resources, causing a circular relationship.

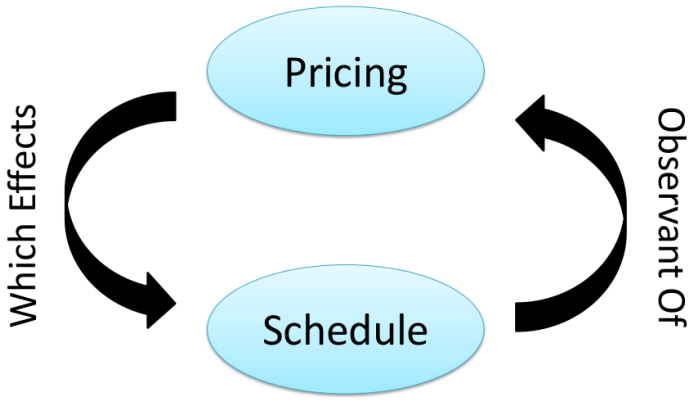

Figure 22: Price and Schedule Dependence

- The schedule has work removed from it which may affect the resource price, though this depends upon the pricing strategy used.

6.3.1 Maintaining Selection Pressure

If this relationship is to be maintained, in order to get prioritisation effects then the price must feature as a selecting pressure in the schedule. If it does not then the price effects the schedule relationship is diminished or removed while the price will remain observant of the schedule.

This relationship may diminish if the budget is set sufficiently high as to no longer be a selection pressure. Noting that the budget being to high is in relation to the mark-up and amount of work that is part of the job i.e. more work or higher mark-ups means the budget must also be higher given the same resource price. In considering the user's preference towards saving money, it is likely that they are going to render as little money available for the job as possible, so they are incentivised to maintain the price effects the schedule relationship.

6.3.2 Effectors of Price Stability

In terms of price stability heterogeneity is an important aspect this can either relate to the jobs, aspects of the schedule or the resources used. In terms of jobs the following aspects cause effects:

Size of the Job (Amount of Work) If the price is set by how far ahead of time the last work completes is then the following is the case. The greater the size of the job, in terms of how much is to be computed then the greater the resource price. If billing occurs with larger jobs then the comparison is performed against the current time so the larger jobs no longer affect the price. This is beneficial as the price is gradually becomes

less dependent upon a single job. This mechanism prevents the price from dropping rapidly but when new jobs arrive the price can still spike, although admittedly in doing so it follows demand.

Size of the Job (Task Count) This directly effects how many actions are placed in the queue. If pricing is based upon an action count then during billing actions are removed from the schedule and the price drops. Hence when jobs have many tasks the price drops further, making this form of pricing subject to heterogeneity in the job. The use of action counts in this case also does not respect the relative size of work, so again causes difficulties.

Billing Frequency/Rate Work is Cleared from the Schedule The frequency of billing is of importance, in cases where the removal of work from the schedule causes changes in the price. This can be seen in cases such as where prices are based upon action/job/task counts, or amount of work in the schedule in terms of cycles, if completed work that is not billed contributes towards the current price.

Billing Frequency If billing is performed at the end of each job more work is cleared from the schedule then if it was billed at the completion of every task. In terms of stability it therefore may be beneficial to bill on a per task basis, even though this breaks the initial premise of billing for a bag of tasks.

Job Acceptance Frequency which is in part upon the arrival rate of jobs, i.e. if no jobs are submitted none may be accepted. It is the principle actor in terms of increasing the current resource price and the greater the amount of work accepted at any one time then the more the price will rise.

It is also based upon parameters such as the remaining time available for the job, i.e. due date and deadline and the budget requirements and size of the job. Hence the overlapping of jobs in regards to their available space should be considered such as in section 6.2.3.

The job acceptance frequency is important as the more work in a given schedule then the greater the price. The acceptance rate should also relate strongly to the ability to clear work from the schedule i.e. over the long term accepting more work than can be cleared is unsustainable as wait times increase indefinitely. Therefore in terms of resources the following aspect also causes effects:

Machine Availability and Speed This relates directly to the ability of the system as a whole to both accept work, i.e. the permitable queue size is bigger and also the ability to clear work away from the schedule is faster.

Network Speed this relates to how quickly data may be transferred and hence the ability to clear and bill work from the schedule. The faster this can

happen the greater the possibility for price fluctuations. In been able to clear work away faster this changes how fast work may be accepted.

The Schedule's Quality This is variation upon the comment of machine availability, speed and network speed etc. It however relates to the schedule generated. If the schedule utilises the resources more efficiently then it is more likely to be able to clear work from the schedule and cause billing events to occur, which in turn changes the price.

6.3.3 Counteracting Price Instability

In sections 6.2.1, 6.2.2 and 6.2.4 it was seen that before the budget constraints became dominant that selection pressures did not favour jobs based upon the budget available. If a Grid become overworked then the economic selection pressures are required. This means the economic factors should be made to be dominant by changing the resource/service price.

It was also seen in section 6.2.2 how price stability affected the ability of the mechanism to maintain this selection pressure. This was then further discussed in section 6.2.3 and assessed further in section 6.2.4. The various ways that might be used to obtain a more stable price are therefore discussed and the possible options are hence listed below:

Step Change From a Previous 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 [1]. 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 [8, 44].

Do not Rely upon Completed Work for Pricing The is by far the simplest solution, by simply ensuring that pricing does not rely upon already completed work for determining the resource cost. This means any billing event will not affect the price, by removing jobs from the schedule 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.

Maintain an Even Billing Distribution An alternative though needlessly complicated option would be to ensure the billing events were purposefully held apart. In holding these events further apart it would reduce the number of occurrences, where a lot of work is removed from the schedule at the same time. This would require a scheduling algorithm that was geared specifically towards the pricing mechanism. If individual tasks were not 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.

6.4 Time and Cost Constraints Compared

Following the experimentation in this section it is possible to compare time and cost constraints and show how they are related within the pricing model, these differences and similarities are therefore explored below.

The budget constrains across all jobs evenly based upon the current system load, regardless of if they can be allocated in a position that is far into the future. This is in difference to due date and deadline which will not constrain based on load if the current load is high and the due date/deadline is sufficiently far in the future. Essentially the closer the temporal parameters are to the current time then the lower the available slack in scheduling and hence the more likely it is that they respond to the current system load. The budget as a selection pressure also adapts based upon the mark-up, current system load and the expected resource usage as indicated by the schedule covering work in the future. Thus if more work is required to be performed then it is more expensive and hence more likely to act as a selection pressure.

The due date and deadline acts as a selection pressure based upon the current load and amount of work to be performed, but does not directly take a mark-up as a modifier. The mark-up however changes the breakeven point which is between the due date and deadline, thus the temporal pressures can be adjusted as well. This effect is not however seen with other [2, 3, 16, 17, 6] related gradient decent based models and is due to the removal of the direct relationship between the available budget and the price paid, causing the introduction of a mark-up for the broker, e.g. in LibraSLA[17] the budget is the price paid unless the soft deadline is passed.

Temporal constraints within the ISQoS system and other related models are also likely to cause a sequence of loading and unloading. This is caused by the pressure of such a time constraint becoming more effective under heavy load and then waning after it has been effective in preventing jobs from entering the market and can be considered symptomatic of deadline based approaches. The budget constraint is however more complex in this regard and can be used in this situation to preferentially select jobs with different budgets, job sizes and mark-ups.

The budget in the experimentation presented acts like a spot price. The price is determined from the immediate load profile. If the due date is far in advance a futures price might be more applicable, as it no longer reflects the current load. This means the price in the future could be different to a more current price and be based upon the estimated future demand!

The ISQoS negotiation mechanism acts as a tender contract market. An auction and batch based system seems incompatible with the idea of rescheduling to meet time constraints. Auctions still have attractive properties in terms

of competition, which is something ISQoS finds more difficult. This discussion is summarised in Table 1.

Table 1: Cost and Time Constraints Compared

Concept	**Due Date/Deadline**	**Budget**
Reflects current load	Yes: unless constraints are far in the future.	Yes: unless budget is very high. Requires price to follow demand.
Influencing factors	None: effects gradient in due date to deadline period, though the breakeven point moves.	Mark-up: Makes the job more favourable to the broker, but consumes the budget quicker.
Constraining upon reflects the job's expected resource usage	Partially: Only when due date and deadline are close to the current time relative to the amount of work required to be performed. Slack however reflects usage.	Yes: Cost increases with more time upon resource that is requested.
Creates availability window (Temporally)	Yes: By Definition.	Yes: If price increases due to load then the budget creates a notion of maximum acceptable load. The work already present removes space earlier on in the schedule.
Creates availability window (Fiscally)	Yes: A breakeven point is formed, between the due date and deadline.	Yes: By Definition.

7 Feature Comparison

In this section the ISQoS broker is compared and contrasted with similar schedulers/brokers. The Grid schedulers and brokers that are evaluated that form the current state of the art are: Condor/G [45], PBS [25], Oracle Grid Engine [46], Globus Toolkit, GridWay [47], OpenCCS [48, 49, 50], Nimrod/G [51] and GridEcon [30, 31]. An initial discussion of the evaluation criteria will be given followed by a tabulated assessment of the criteria.

7.1 Cost & Time Guarantees

In Grids given the finite amount of resources if time guarantees are to be implemented then it is required to have an understanding of the priority of each job

to be completed as only a certain fraction of jobs can be completed within the allotted time constraints. Economics is a technique that may be used to realise this prioritisation.

The ISQoS Broker and similar brokers such as Nimrod/G have been developed with economics in mind. Nimrod/G provides the choice of four different scheduling algorithms and allows for the selection of cost and time constraints [20]. The ISQoS providers allow for variable scheduling algorithms to be used, both economically aware and otherwise. In ISQoS it is possible for cost unaware scheduling algorithms to be used within the process while maintaining an economic approach due to the economic models structure. This approach is preferred as it allows ISQoS providers to return estimates as guidance, when the scheduling algorithm cannot meet the economic or time constraints, which is not possible when following economically oriented algorithms such as in Nimrod/G.

The Nimrod/G cost algorithms are simple as previously discussed in section 5.2. For example if the cost optimize algorithm is used and the deadline constraint is not actively having an affect then it stacks all work upon the cheapest resource, without any form of load levelling. This is even the case if a second equally cheapest resource is present. The Cost Time Optimization algorithm partly gets around this by grouping resources of equal cost together and then allocating to the fastest resource, thus providing better load balancing. Though this is not best when dealing with costs on a continuous scale, where using the ratio between cost and resource speed would be better. The time optimization algorithm works in a simple earliest completion time fashion while respecting the budget constraints.

Oracle Grid Engine has three classes of job ranking policy, including a ticket based mechanism that provides a proportional share based mechanism, an urgency based mechanism i.e. priority and the final option is to have custom policies [46]. Proportional share mechanisms can be seen to be a primitive economic mechanisms [44] hence Oracle Grid Engine could be considered to have some cost awareness. Though proportional share based mechanisms tend only to consider sharing resources and do not further control the flow of where jobs are allocated.

7.2 Advance Reservation Support

Advance Reservation is a key requirement in been able to establish any form of completion time guarantees as it provides the opportunity for guarantees to be realised.

Advance reservation is possible in several middleware solutions, but it must be noted at that it requires estimated durations for each task, which are not always available. This therefore means that reservations must last at least as long as the duration of the task for the execution to work correctly. Middleware is hence is likely to support reservations but is not likely to put them to use in environments centred upon having high machine utilisation.

7.3 Queuing vs Scheduling

Most Grid Infrastructure (schedulers/brokers/middleware) use queue based mechanisms as opposed to scheduling in the same fashion as operational research. Condor, PBS and Grid Engine are good examples of such a queue based approach. The requirement to act in a queue based fashion, stems from the need to keep the resources of the Grid working and no intent to provide time and cost provision. A scheduling based approach requires that job's size is known, but it then allows for guarantees in completion time to provided. The Mauri Scheduer, OpenCCS and ISQoS mechanisms differ in this regard and use a scheduling based approach. Below in Table 2 the differences between a queuing and scheduling based system are highlighted [49]:

Table 2: Queuing vs Scheduling [49]

	Queuing System	**Scheduling System**
Considered time frame	present	present and future
Runtime estimates	not required	required
Submission leads to	insert in queues	complete reschedule
Job start times known	no	yes
Reservations	difficult	yes, trivial
Deadline scheduling	optional	yes, trivial
Backfilling	optional	yes, trivial
Examples	PBS, NQS, LL, ...	Maui Scheduler, OpenCCS

7.4 Negotiated Job Submission

In regards to job submission it is useful to understand how quickly results are likely to be returned from each provider. This is dependent upon the current workload of the environment relative to its capacity and other constraints such as the job's priority/value and the jobs size.

In order to achieve an estimated completion time, a candidate schedule needs producing. In other none ISQoS infrastructures this mechanism simply does not exist. Jobs are simply submitted to a provider by a user whom has been informed of current resource utilisation.

The lack of this candidate schedule mechanism limits the ability to perform any sense of negotiation as it would be difficult to relate the work that is to be submitted and its completion requirements to resource availability. It also means brokers are reliant upon metrics such as current workload and queue length which are unrealistic measures to use when guarantees upon cost and time of job completion are required.

Current workload (i.e. amount of machines/operators that are busy) and queue length measures only relate to the current situation upon the Grid, which will quickly change. They could be likened to a call centre in which a blanket statement of all operators are busy and you are x in the queue. In the call

centre's case many "jobs" are of a similar size and the expected time to been served can be guessed. In the case of the Grid jobs differ in sizes and have varying amounts of data to transfer which causes coupling between the jobs i.e. two jobs using the same network can only stage out/in together by sharing the bandwidth, thus slowing each other down.

7.5 Monitoring, Adaptation and Rescheduling

Rescheduling is an important tool when the dynamic nature of the Grid is considered. It can aid in recovering from errors and is also useful in moving tasks' advance reservations in order to honour more agreements or the current agreements better.

A summary of the capacity to reschedule is provided below:

- ISQoS - Periodic scheduling, with the ability to dynamically select the period at runtime and rescheduling upon job submission is also allowed.
- Oracle Grid Engine - Periodic scheduling only, jobs enter a queue before being scheduled fully [46].
- GridWay MetaScheduler - It has both periodic and event driven rescheduling i.e. when performance slowdown or remote failure is detected [52].
- OpenCCS Rescheduling is performed upon the submission of a new job.
- Condor The DAG (Directed Acyclic Graph) manager that handles workflows is capable of generating rescue DAGs when it exists with an error code. The rescue DAG is a new DAG listing the elements of the original DAG left unexecuted. To remedy the situation, the user can examine the rescue DAG correct any mistakes in submission and then resubmit it as a normal DAG.

7.6 Advanced Discovery, Scheduling API & Flexible Pricing

In the ISQoS provider delivers an API for developers of new scheduling algorithms, it is capable of hiding away much of the difficulty of Grids and presents the scheduling as merely the allocating of jobs to resources. For example it determines the estimated completion time for each job on a given resource and uses the same API mechanisms for both execution and data transfers. The advanced discovery mechanism given presents a list of capable resources for a given job that are available for allocation, thus removing an implementers requirements to check themselves.

The scheduling algorithm may easily be swapped out for experimental purposes (see section 5.2) with alternatives and the pricing mechanisms may also similarly be swapped out (see section 5.4) such that any commodity market pricing strategy can be formed by a given provider.

7.7 User vs System Centric Scheduling

The majority of Grid Middleware/Brokers and Schedulers are system centric[11, 4, 2, 53]. In that they aim to maximise utilisation of Grid resources or are otherwise orientated towards system oriented metrics.

ISQoS is different in its setup in that it is more balanced and considers the users objectives, especially in terms of delivery time and how much it costs. The cost however is also system centric in that the price can be made to adapt to current load and hence control utilisation, whilst also ensuring the provider benefits from providing Grid resources.

7.8 Open Standards

In Grid middleware in order to gain a degree of interchangeability and general compatibility so that different middleware's can work together either open standards or a large amount of adaptors, that complicate integration efforts are needed. In recent years there has been a heavy drive towards open standards. The ISQoS broker therefore implements standards such as WS-Agreement, WS-Agreement negotiation and Job Submission Description Language (JSDL) as well as web service based standards, WSRF, WS-Notification, WS-Security.

7.9 Comparison Study

In this section each of the brokers/middleware are compared. They are split into three tables, one for best effort approaches (Table 3), one for QoS oriented (Table 4) and one for economic approaches (table 5).

In Table 3 the best effort approaches are discussed. These are characterised by queue based scheduling that is centred around the resources and aiming to keep them as busy as possible.

In later work there are substantial efforts to provide quality of service provision with respect to reliability. The vast variations of different middleware and their configurations gives rise to meta-schedulers such as GridWay, i.e. ones that control and submit work to local Grid infrastructures using adaptors, which in turn schedule the work for completion. The OpenCCS software provides a fully schedule based solution to deploying Grid resources and given the work of the HPC4U project allows for resilience to resource failure. These two QoS oriented middlewares are shown in Table 4.

There is also a focus upon the development of economic models to control Grids, given their diverse nature. Early work such as Nimrod/G focussed solely upon Auction based approaches. Later work such as GridEcon took a top down approach creating a set of economically oriented services which be used in the Grid. The ISQoS project takes this service orientated approach and develops quality of service around the concept of job completion time and cost. These economic approaches are shown in Table 5.

Table 3: Feature Comparison Study - Best Effort

	Best Effort			
	Condor-G	Portable Batch System (PBS)	Oracle Grid Engine	Globus
Advanced Discovery	no	no	no	yes via Globus MDS
Cost & Time Guarantees	no	no	no	no
Advance Reservation Support	no	no/yes with Maui	yes	yes
Queuing vs Scheduling	queue	queue scheduling with Maui	queue	queue
Negotiated Job Submission	no	no	no	no
Rescheduling	no	no	allows checkpointing	no
Monitoring	yes	no	some	yes
Adaptation	no	no	some - QMaster can auto-restart	no
Open Standards	no	no	some DRMAA	no
User vs System Centric	system	system	system	system
Plug-in Scheduling	no	no	yes (some flexibility)	no
Flexible Pricing	no	no	no	no

Table 4: Feature Comparison Study - QoS

	QoS	
	OpenCCS	GridWay
Advanced Discovery	yes	yes
Cost & Time Guarantees	time only	no
Advance Reservation Support	yes	depending on underlying infrastructure
Queuing vs Scheduling	queue	queue
Negotiated Job Submission	no	no
Rescheduling	yes with HPC4U	yes
Monitoring	yes with HPC4U	yes
Adaptation	yes with HPC4U	yes
Open Standards	no (WS-Agreement used in AssessGrid)	some DRMAA
User vs System Centric	system	system
Plug-in Scheduling	yes	yes
Flexible Pricing	no	no

Table 5: Feature Comparison Study - Economic

Economic			
	Nimrod/G	GridEcon	ISQoS
Advanced Discovery	yes	potential	yes
Cost & Time Guarantees	yes	potential	yes
Advance Reservation Support	no	potential	yes
Queuing vs Scheduling	scheduling	scheduling	scheduling
Negotiated Job Submission	yes	potential	yes
Rescheduling	yes	potential	yes
Monitoring	no	potential	yes
Adaptation	no	potential	potential
Open Standards	no	yes	yes
User vs System Centric	system	potential	both
Plug-in Scheduling	no	yes	yes
Flexible Pricing	no auction only	yes	yes

8 Conclusion

This paper presents the ISQoS broker and its associated economic model for applying guarantees to jobs on the Grid in terms of time and cost. The model creates upper bounds around these two constraints in a graduated fashion in order to discourage the approach to such constraints. The SLA structure and negotiation process is discussed that allows the user to understand through rounds of negotiation the current status of the Grid with a reflection upon the work that is being submitted.

An evaluation is performed that demonstrates how the proposed model generates selective prioritising behaviour based upon the budget. This is performed with SLAs on a per jobs basis, under the understanding that it assists prioritising individual jobs and provides a clear route for pricing based upon existing workload pressures.

It is then further shown that rescheduling can effect the market by reducing the price rapidly by billing events in rapid succession. The effect of which removes actions from the schedule that are in turn used as a measure of the current load which determines the price. Recommendations are hence made to alleviate the issues encountered and demonstrated. This can be seen as an introduction into how scheduling can have market implications and further study of different algorithms and their markets effects is considered to hold merit.

The paper is then summarised with a discussion upon the differences between the time and cost constraints within such gradient based time cost models and a comparison study is performed between the ISQoS broker and other selected work from the literature. The work presented has been shown to be a viable mechanism for providing time and cost guarantees by negotiation within a compute service market, that is equally applicable to both Grids and Clouds acting as a compute service, where payment is associated with the work performed and not so directly the tool/virtual machine that was used to perform the work.

We would like to thanks the Engineering and Physical Science Research Council (EPSRC) for funding the Intelligent Scheduling for Quality of Service (ISQoS) research project (EPSRC: Reference EP/G054304/1) from October 2009 - March 2013, thus making our work possible.

References

[1] Iosup A, Epema D. Grid computing workloads. *Internet Computing, IEEE* 2011; **15**(2):19–26, doi:10.1109/MIC.2010.130.

[2] Chun BN, Culler DE. User-centric performance analysis of market-based cluster batch schedulers. *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, 2002; 30–30.

[3] Irwin DE, Grit LE, Chase JS. Balancing risk and reward in a market-based task service. *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, 2004; 160–169.

[4] Buyya R, Abramson D, Venugopal S. The grid economy. *Proceedings of the IEEE* 2005; **93**(3):698–714, doi:10.1109/jproc.2004.842784.

[5] Kavanagh R, Djemame K. Negotiated economic grid brokering for quality of service. *4th FTRA International Conference on Computer Science and Its Applications, CSA 2012*, vol. 203 LNEE, Springer Verlag: Jeju Island, Korea, Republic of, 2012; 87–96, doi:10.1007/978-94-007-5699-1_10. URL http://dx.doi.org/10.1007/978-94-007-5699-1_10.

[6] AuYoung A, Grit L, Wiener J, Wilkes J. Service contracts and aggregate utility functions. *15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15)*, IEEE: New York, 2005.

[7] Kokkinos P, Varvarigos EA. A framework for providing hard delay guarantees and user fairness in grid computing. *Future Generation Computer Systems* 2009; **25**(6):674–686.

[8] Neumann D, Ster J, Weinhardt C, Nimis J. A framework for commercial grids - economic and technical challenges. *Journal of Grid Computing* 2008; **6**(3):325–347.

[9] Merlo A, Clematis A, Corana A, Gianuzzi V. Quality of service on grid: architectural and methodological issues. *Concurrency and Computation: Practice and Experience* 2011; **23**(8):745–766, doi:http://dx.doi.org/10.1002/cpe.1641.

[10] Fox G, Ko SH, Pierce M, Balsoy O, Kim J, Lee S, Kim K, Oh S, Rao X, Varank M, *et al.*. Grid services for earthquake science. *Concurrency and Computation: Practice and Experience* 2002; **14**(6-7):371–393.

[11] Buyya R, Abramson D, Giddy J, Stockinger H. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14**(13-15):1507–1542, doi:http://dx.doi.org/10.1002/cpe.690.

[12] Vickrey W. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance* 1961; **16**(1):8–37, doi:10.1111/j.1540-6261.1961.tb02789.x.

[13] Schnizler B, Neumann D, Veit D, Weinhardt C. Trading grid services - a multi-attribute combinatorial approach. *European Journal of Operational Research* 2008; **187**(3):943–961.

[14] Schnizler B, Neumann D, Veit D, Weinhardt C. A multiattribute combinatorial exchange for trading grid resources. *Proceedings of the 12th Research Symposium on Emerging Electronic Markets (RSEEM)*, 2005.

[15] Wilkes J. Utility functions, prices, and negotiation. *Market Oriented Grid and Utility Computing*, Buyya R, Bubendorfer K (eds.). Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, Inc, 2008; 67–88.

[16] Popovici FI, Wilkes J. Profitable services in an uncertain world. *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, 2005; 36–36.

[17] Chee Shin Y, Buyya R. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. *Cluster Computing, 2005. IEEE International*, 2005; 1–10.

[18] Kertsz A, Kacsuk P. A taxonomy of grid resource brokers. *Distributed and Parallel Systems*. 2007; 201–210, doi:http://dx.doi.org/10.1007/978-0-387-69858-8_20.

[19] Biette M, Voss K, Padgett J, Gourlay I, Djemame K, Fally B, Ponsard C, Mouton S, Stmke J, Ttard F. Preliminary exploitation plan version 1.0 2006.

[20] Buyya R. Economic-based distributed resource management and scheduling for grid computing. PhD Thesis, Monash University, Melbourne, Australia 2002.

[21] Venugopal S, Xingchen C, Buyya R. A negotiation mechanism for advance resource reservations using the alternate offers protocol. *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, 2008; 40–49, doi: 10.1109/iwqos.2008.10.

[22] Buyya R, Venugopal S. The gridbus toolkit for service oriented grid and utility computing: an overview and status report. *Grid Economics and Business Models, 2004. GECON 2004. 1st IEEE International Workshop on*, 2004; 19–66, doi:10.1109/GECON.2004.1317583.

[23] Xingchen C, Nadiminti K, Chao J, Venugopal S, Buyya R. Aneka: Next-generation enterprise grid platform for e-science and e-business applications. *e-Science and Grid Computing, IEEE International Conference on*, 2007; 151–159, doi:10.1109/e-science.2007.12.

[24] Middleton S, Surridge M, Benkner S, Engelbrecht G. Quality of service negotiation for commercial medical grid services. *Journal of Grid Computing* 2007; **5**(4):429–447, doi:http://dx.doi.org/10.1007/s10723-007-9080-x.

[25] Bode B, Halstead D, Kendall R, Lei Z, Hall W, Jackson D. The portable batch scheduler and the maui scheduler on linux clusters. *Proceedings Of The 4th Annual Linux Showcase And Conference*, Usenix Association: Atlanta, 2000; 217–224.

[26] Jackson D, Snell Q, Clement M. Core algorithms of the maui scheduler. *Job Scheduling Strategies for Parallel Processing*. 2001; 87–102.

[27] Cao J, Zimmermann F. Queue scheduling and advance reservations with cosy. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004; 63, doi:10.1109/ipdps.2004.1302989.

[28] Keller A, Ludwig H. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management* 2003; **11**(1):57–81, doi:10.1023/a:1022445108617.

[29] Han Y, Youn CH. A new grid resource management mechanism with resource-aware policy administrator for sla-constrained applications. *Future Generation Computer Systems* 2009; **25**(7):768–778.

[30] Altmann J, Neumann D, Fahringer T, Courcoubetis C, Stamoulis G, Dramitinos M, Rayna T, Risch M, Bannink C. Gridecon: A market place for computing resources. *Grid Economics and Business Models*, vol. 5206. Springer Berlin / Heidelberg, 2008; 185–196, doi:10.1007/978-3-540-85485-2_15.

[31] Altmann J, Courcoubetis C, Darlington J, Cohen J. Gridecon the economic-enhanced next-generation internet. *Grid Economics and Business Models*. 2007; 188–193, doi:http://dx.doi.org/10.1007/978-3-540-74430-6_17.

[32] Djemame K, Padgett J, Gourlay I, Armstrong D. Brokering of risk-aware service level agreements in grids. *Concurrency and Computation: Practice and Experience* 2011; **23**(13):1558–1582.

[33] Iosup A, Sonmez O, Anoep S, Epema D. The performance of bags-of-tasks in large-scale distributed systems. *Proceedings of the 17th international symposium on High performance distributed computing*, ACM: Boston, MA, USA, 2008, doi:http://doi.acm.org/10.1145/1383422.1383435.

[34] Buyya R, Giddy J, Abramson D. An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications. *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.

[35] Hudert S, Ludwig H, Wirtz G. Negotiating slas-an approach for a generic negotiation framework for ws-agreement. *Journal of Grid Computing* 2009; **7**(2):225–246, doi:10.1007/s10723-009-9118-3.

[36] Open Grid Forum. Job submission description language (jsdl) specification, version 1.0 2005.

[37] Burke S, Andreozzi S, Field L. Experiences with the glue information schema in the lcg/egee production grid. *Journal of Physics: Conference Series* 2008; **119**(6, 062019).

[38] Ganglia Project. Ganglia monitoring system 2012. URL http://ganglia.sourceforge.net/.

[39] Buyya R, Murshed M. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14**(13-15):1175–1220, doi:http://dx.doi.org/10.1002/cpe.710.

[40] Buyya R, Murshed M, Abramson D, Venugopal S. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. *Software: Practice and Experience* 2005; **35**(5):491–512, doi:http://dx.doi.org/10.1002/spe.646.

[41] Kavanagh R, Djemame K. A grid broker pricing mechanism for temporal and budget guarantees. *8th European Performance Engineering Workshop (EPEW'2011)*, vol. 6977, Thomas N (ed.), Springer: Borrowdale, The Lake District, UK, 2011.

[42] OpenNebula Project. Opennebula homepage 2012. URL http://opennebula.org/.

[43] Citrix Systems. Home of the xen hypervisor 2012. URL http://www.xen.org/.

[44] Lai K. Markets are dead, long live markets. *SIGecom Exch.* 2005; **5**(4):1–10, doi:10.1145/1120717.1120719.

[45] Wisconsin-Madison UO. Condor manual version 7.2.4 2009. URL http://www.cs.wisc.edu/condor/manual/.

[46] Templeton D. Beginner's guide to sun grid engine 6.2 installation and configuration white paper 2009.

[47] Distributed Systems Architecture Group. Documentation 2009. URL http://www.gridway.org/doku.php?id=documentation.

[48] Computing PCFP. Openccs user manual version 0.9.1 2012.

[49] Battre D, Hovestadt M, Kao O, Keller A, Voss K. Planning-based scheduling for sla-awareness and grid integration. *PlanSIG 2007 The 26th workshop of the UK Planning and Scheduling Special Interest Group*, vol. 1, Bartk R (ed.), Prague, Czech Republic, 2007; 8.

[50] Battre D, Hovestadt M, Kao O, Keller A, Voss K. Virtual execution environments for ensuring sla-compliant job migration in grids. *IEEE International Conference on Services Computing SCC '08.*, vol. 2, 2008; 571–572, doi:10.1109/SCC.2008.106.

[51] Monash eScience and Grid Engineering Laboratory. The nimrod toolkit 2009. URL http://messagelab.monash.edu.au/Nimrod.

[52] Huedo E, Montero RS, Llorente IA. A modular meta-scheduling architecture for interfacing with pre-ws and ws grid resource management services. *Future Generation Computer Systems* 2007; **23**(2):252–261, doi:10.1016/j.future.2006.07.013. Futur. Gener. Comp. Syst.

[53] Vanmechelen K, Broeckhove J. A comparative analysis of single-unit vickrey auctions and commodity markets for realizing grid economies with dynamic pricing. *Grid Economics and Business Models*, vol. 4685, Veit D, Altmann J (eds.). Springer Berlin / Heidelberg, 2007; 98–111, doi:10.1007/978-3-540-74430-6_8.