SLA Based Job Scheduling: A Case Study on Policies for Negotiation with Resources

Viktor Yarmolenko Rizos Sakellariou (first.last@manchester.ac.uk)
School of Computer Science, University of Manchester, M13 9PL, UK

Djamila Ouelhadj Jonathan M Garibaldi (dxs, jmg @cs.nott.ac.uk)
School of Computer Science and IT, University of Nottingham, NG8 1BB, UK

Abstract

In this paper, we present work on an EPSRC e-Science project whose purpose is to investigate the effects of SLA based approaches for parallel computing job scheduling. We considered a coordinator based architecture, where the Coordinator forms SLAs with Users and Resources (parallel servers). The paper focuses on a case study which evaluates different policies for the negotiation of SLAs between the Coordinator and the Resources. We show that the use of information available in the SLAs agreed by the User may fundamentally alter the behaviour of the Coordinator and hence the performance of the overall system. These effects need to be understood more deeply before new designs can take advantage of the possibilities offered by SLAs.

1 Introduction

Traditionally, the enactment of jobs on parallel supercomputer resources has been based on queue-based scheduling systems, offering only one level of service, namely 'run the job when it gets to the head of the queue'. New patterns of usage have resulted in the introduction of advance reservation; however, this is an extreme level of service again since it presupposes a precise time to run a job is available. It is only recently that work [1, 2, 3] began to understand how more flexible approaches, based on *Service Level Agreements* (SLAs) could be effectively used for job farming. The new avenues which SLAs might open to the design of job management, and in particular brokering and scheduling, are still poorly explored.

The aim of this paper is to present some of the work in the EPSRC project "Service Level Agreement Based Scheduling Heuristics" (funded by the EPSRC Fundamental Computer Science for e-Science initiative through Grants GR/S67654/01 and GR/S67661/01), whose purpose is to understand and provide a quantitative analysis of the main aspects of an SLA based management system for parallel computing job scheduling.

The next section outlines published work relevant to this paper. Section 3 describes the key features of the model architecture used, namely topology, negotiation schema and metadata. The design of an experiment aiming to evaluate different policies for the negotiation of SLAs with local resources is presented in Section 4. Results are presented in Section 5 followed by a conclusion.

2 Related Work

An SLA can be described as a contract between participants, outlined as a set of guarantees, QoS metrics and behaviours and should be treated as a legal document of the transaction. In the short paper [1] from Fermi Lab (IL, USA) Hathaway presents a list of SLA attributes and their importance. Important questions are raised, such as "Why need SLA?", "With whom SLA should be established?", "How do we determine the level of service?". The paper also presents a list of basic components that comprise an SLA. Later on, several focus groups were formed to work on standards. The main contributions where WS-* [2], which concentrates on agreement structure and specification, whereas attempt to standardise protocols for negotiating and allocating resources on Grid were made in SNAP [4] and GRAAP [5] respectively. All this work is still in progress. Recently, Padjett *et al.* tailored WS-A spec to build a management architecture that monitors and validates SLA in Grid. The architecture was designed to satisfy a specific need of the DAME project [6] and the SLA included various logs for job tracking as well as standard Service Level Objectives (SLO) introduced in WS-A.

Client (User or Coordinator)		Service (Coordinator or Resource)
Request a service (job, SLO)	$\rightarrow$	Request is received, decision is formed
Quote received, decision is made	$\leftarrow$	Sending the quote
Agreement is reached (or not)	$\rightarrow$	SLA is Recorded

Table 1: The single negotiation sequence of type (1).

The use of SLAs in the domain of computational resources is not completely unchartered. For example, a Grid resource SLA broker is presented in [7] - GRUBER, in which resource usage policy (SLA) is separated from resource access policy. One of the GRUBER's four components is, surprisingly, a queue manager - a complex client that resides on submitting machine and monitors policies, decides how many jobs to submit where and when. However, the use of SLAs (the syntax of which was based on [8, 9, 10]) has not been taken into account when scheduling. Instead, fair share scheduling [11] techniques were employed. The broker's site selection strategies also did not take advantage of the SLA.

Work done in the domain of bandwidth scheduling using SLA has advanced further than SLA job scheduling. A concrete framework on SLA based optimisation was constructed with regards to network resources, IP Services [12], Web Services [13]. In the latter, the authors use the concept of virtualized resources. They present a detailed design of the architecture and some performance results. Yun Fu *et al.* proposed an SLA based scheduling approach of distributed resources for streaming [14] - SLASH. They explored price and penalty model for SLA and suggested a dynamic scheduling algorithm - Squeeze - which takes into account prices and penalties charged for the distributed resource. Other related work has been presented in [15, 16, 17].

3 Model Architecture

During the past years, research in Grid middleware for job and resource management has led to different designs and implementations which vary in complexity and functionality. One of the key features of any such design is how light is the user client; in other words, which module controls what aspect of the job submission and scheduling. This partly determines the system's topology, negotiation mechanisms and the metadata that needs to be transfered.

3.1 Topology

In this paper we assume a centralised topology in which all jobs are submitted through a Coordinator (or Broker). Users agree an SLA with the Coordinator regarding their jobs. In turn, the Coordinator agrees (independently) an SLA for each job with one of the parallel servers. This is schematically shown in Figure 1.

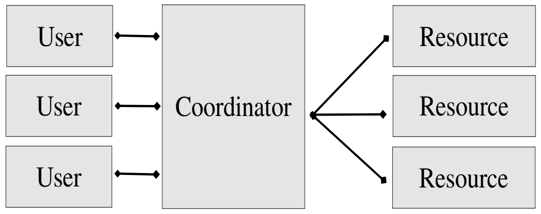

Figure 1: The centralised model. SLA is formed between User and Coordinator independently of the agreement between Coordinator and Resource.

3.2 Negotiation

In Section 2 we mentioned some of the work done on negotiation. Having considered these we can logically categorise all negotiations into three types: (1) Bid by Server, (2) Bid by Client and (3) Matchmaking. Note, that conventional buying is simply a special case of bidding in which the first offer is accepted. The first type of negotiation involves services bidding for the business suggested by the client. This business model is well studied and widely implemented in customer to business relationship. The service regulates itself based on the customer's demand. Negotiation type (2) is relatively young and involves clients biding for the service, say CPU time. A number of web based auctions such as eBay [18] adopt such a model where clients are bidding for the service/product, usually constrained by the set of parameters (such as deadline, location, payment method, *etc.*). Type (3) is the least exploited in contemporary business. It involves a third party to be trusted with production of a successful agreement based on requirements and preferences of all parties (both client and server in a simplest case). This negotiation model may work well with certain SLA structures and topologies, but to be effective harsh demands must be imposed on a module that is handling the negotiation.

In this paper, we assume a 'bid by Server' protocol (type 1), which we simplified to a bare minimum (Table 1); we also assume it is used for both

Context	Comment	SLO	Comment
SLA ID	Unique identifier of SLA	T_S	Earliest start time for the job
Job ID	Unique identifier of a Job	T_F	Latest finish time for the job
Participants	List of participants of this SLA	t_D	Job Duration (provided by the client)
	(e.q. Coordinator, Resource,..)	N_{CPU}	The number of CPU Nodes required

Table 2: Table of SLA Parameters: Context and Service Level Objectives (SLO).

types of negotiations User to Coordinator and Coordinator to Resource. First the client (User or Coordinator) contacts the service (Coordinator or Resource respectively) with the request. The contents of the request are identical to those outlined in Table 2 (SLO section). The service forms a decision (either independently, as it is the case in the experiment presented in this paper, or negotiates the service further down the chain) and responds with an answer. The positive answer would contain a quote for the requested service (effectively an SLA signed by the service side only). If the quote is acceptable to the client it, in turn, signs the quote - an SLA is formed. The model allows for the unlimited number of negotiations made before the successful SLA is formed.

3.3 SLA Specification

The negotiation strategies are partially chosen because of the view of the desired outcome - the agreement (SLA). Therefore it is vital to choose the SLA specification such that to optimise the cost of formed SLA (i.e. negotiation time, size of metadata, *etc.*)

The idea to use SLA to record guarantees of a service after negotiation is completed was a logical step in Grid development, having migrated from realm of B2B commerce and, as such, was around for a while. Since the formation of the relevant focus groups [4, 5], various research projects [6, 7] tailored SLAs to their specific requirements.

In this paper we introduce only few Service Level Objectives (SLO) (Table 2), all of which can map onto the generic WS-Agreement specification [2]. In the project we also aim to work on the extension of WS-Agreement; this is beyond the scope of this paper. What is important is that the SLA in Table 2 contains a set of constraints and additional information so that the overall performance of the system can be evaluated in the context of these additional parameters.

4 Experimental Design

The aim of the experiment is to evaluate the process of the allocation and submission of an arbitrary job on a distributed resource using an SLA not only as an enforcement mechanism but also as a set of constraints that determine the submission process. The model used maps topologically onto the architecture shown in Figure 1. In the experiment we compare traditional strategies with strategies that make use of information in an SLA to steer the job submission process (more on this in 4.2). Further, we describe the specifics of the model as a whole followed by the metrics used to evaluate it, after which we outline important features of each module in the model.

The experiment considered has a varied number of Resources per one Coordinator, $N_{res} = \{2, 4, 8, 16, 32, 64\}$. Each Resource has a capacity of 64 parallel CPUs ($C_{total} = 64$). For each of the N_{res} Resources there was a set of job requests, $N_{job} = 340$, which where generated such that they guarantee that a solution existed whereby the Resource's scheduling mechanisms could potentially utilise 100% of Resource's capacity, within the availability time window of 147 virtual hours, thus resulting in no SLA left unsatisfied and the resources being fully utilised. In other words, for each set up there was a set of $N_{res} \times N_{job}$ jobs that potentially can fit into N_{res} Resources over the time window of 147 virtual hours within the requirements specified by an SLA.

The negotiation process in the model is carried out in the following way. First the User submits a job request (see Section 3.2) to the Coordinator, which immediately forms an SLA with the User. Then, the Coordinator separately negotiates an SLA for each job request with Resource. It sequentially attempts to negotiate with a resource until a resource is found that is willing to make an SLA regarding this job. The aim of our experiments is to evaluate the number of negotiations of different policies.

In addition to those mentioned above, the following assumptions about the model were made:

- Once an SLA between Coordinator and Resource is agreed it will be satisfied.
- The candidate list of Resources is static during the single simulation run and its size is N_{res}.
- All N_{res} resources are always potentially available.
- The general information about Resource's current capacity is available to the Coordinator.

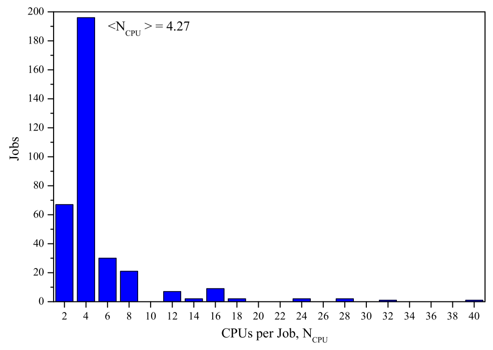

Figure 2: The distribution of CPUs per job, N_{CPU}. $\max(N_{CPU}) = 40$, $\langle N_{CPU}\rangle = 4.27$.

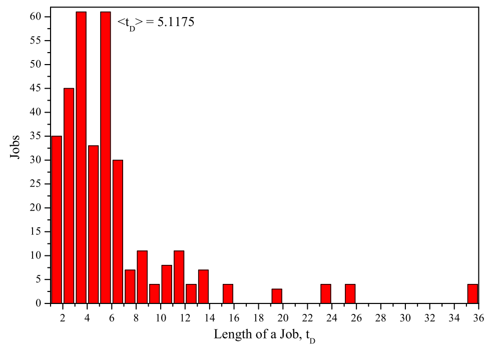

Figure 3: The distribution of the duration of each job, t_D. $\max(t_D) = 35$, $\langle t_D\rangle = 5.12$.

- The negotiation time, t_{neg}, is constant for any single connection made and, thus, is set to 1.
- The number of negotiations per SLA between User and Coordinator is always 1.
- The minimal number of negotiations per SLA that can be achieved between Coordinator and Resource is $\min(N_{neg}) = 1$, whereas the maximum number of negotiations is usually $\max(N_{neg}) = N_{res}$, unless a policy that may attempt to negotiate more than once with the same resource is used.

The standard metrics used in measuring the performance of a job scheduling management system, such as schedule makespan or job start up delay, *etc.* are no longer adequate for the system investigated. For example, in this model there is no concept of a queue as it is common for the batch systems, due to its SLA steered allocation process. Instead, we evaluate it in terms of effectiveness of the system in reaching the 100% SLA satisfaction. This arrangement gives a firm framework in which the model with various parameters can be quantitatively compared.

Let us define the negotiation factor, f_{neg}, as:

$$f_{neg} = \frac{N_{neg} - \min(N_{neg})}{\max(N_{neg}) - \min(N_{neg})} = \frac{N_{neg} - 1}{N_{res} - 1} \quad (1)$$

Here, the negotiation factor is simply the number of absolute negotiation attempts N_{neg} performed before an SLA is achieved. Note, we define $\max(N_{neg})$ as the maximum reasonable number of negotiation attempts, which is equal to N_{res} because one of the Resources is bound to accept the job request (although some policies may exceed this value). In the model, $N_{res} \times N_{job}$ jobs provide 100% potential utilisation for all Resources, therefore let us define the utilisation factor, f_{load}, as a percentage of the entire job pool for which a number of successful SLAs (N_{SLA}) were formed so far, between the Resource and Coordinator:

$$f_{load} = \frac{N_{SLA}}{N_{res} \cdot N_{job}} \cdot 100\% \quad (2)$$

Thus the evaluation of the model will consist in measuring the negotiation factor, namely its dependence on the resource utilisation or load, $f_{neg}(f_{load})$. This dependence is then compared to the system with a different algorithm for Coordinator as well as different number of Resources per Coordinator. Further we discuss three modules of the model investigated in more detail (Sections 4.1-4.3).

4.1 User

The User module generates a set of job requests (containing: N_{CPU}, t_D, T_S, T_F, as denoted in Table 2). The module generates the required percentage of utilisation with specified number of CPUs required per job, N_{CPU}, and the job duration, t_D. In the cases presented we used 100% utilisation. The distribution of CPUs per job in a set N_{res} is shown in Figure 2. The duration of a job t_D is varied between 1 and 35 virtual hours with a step of 1 virtual hour. The overall distribution of t_D per job in a set N_{res} is shown in Figure 3. The general role of the User module is to generate job requests as determined by the required distribution and to negotiate an SLA with the Coordinator.

4.2 Coordinator

In general, the role of the Coordinator can be described in many aspects (as a proxy, as a router, as a broker, as a security gate). The measurement technique can only reflect a limited number of these aspects.

In this experimental study, we are looking at coordinator's ability to effectively minimise the number of single negotiations per SLA achieved. The number of negotiations between Coordinator and Resource could be reduced if the Coordinator intelligently selects its candidates. The negotiation with such candidates would result in a successful SLA with a minimal number of single interactions. In other words, one of the coordinator's roles is to predict the outcome of the request with certain accuracy and, based on that prediction, decide whether to engage in negotiation of SLA. The more effective is the algorithm the more Resources can be handled by a single Coordinator at any given time interval or the less traffic is generated for each SLA.

The overall performance of the Coordinator (and in this case the whole system) can be evaluated in terms of the average time it took to form a successful SLA and is defined in the relation below:

$$\langle t_{SLA} \rangle = \frac{1}{N_{SLA}} \sum_{i=1}^{N_{SLA}} \left(t_{dec}(f^i_{neg}) + t_{neg}(f^i_{neg}) \right) \tag{3}$$

where $t_{dec}()$ and $t_{neg}()$ are the functions that return respectively the time required by Coordinator to form a decision as to which Resource to connect and the time spent on negotiating with that Resource. As pointed out before, it may take more than one negotiation attempt before a successful SLA is formed and functions $t_{dec}()$, $t_{neg}()$ represent the overall time it took to secure an SLA for each job i, hence the dependence on f^i_{neg} (from Equation 1). $\langle t_{SLA} \rangle$ is averaged over the times of all (N_{SLA}) agreements made. $t_{dec}()$ and $t_{neg}()$ have linear dependence in simple case, but expand to non-trivial functions in real life. In our case $t_{neg}(f^i_{neg}) = 1 \cdot N^i_{neg}$ is linear and $t_{dec}(f^i_{neg}) = \sum_{j=1}^{N_{neg}} t_j$ depends on which negotiation attempt it is at (see Table 3). Depending on Coordinator, for any successful SLA formed, the first negotiation is different whilst all consecutive negotiation attempts are equal to each other.

We used four different algorithms for Coordinator. The four different algorithms are:

- RAB - Random Access Basic, in which the Coordinator randomly accesses Resources and does not remember those already visited. Thus, its values for N_{neg} can potentially be higher than $\max(N_{neg})$ as defined earlier (i.e. $f_{neg}() = \{0, \infty\}$).

- RAE - Random Access Exclusive, in which the Coordinator randomly accesses Resources, but excludes the visited Resource after each failed negotiation attempt (i.e. $f_{neg}() = \{0, 1\}$).

- LAF - Least Accessed resource First, in which the Coordinator prioritises Resources in the order of the least accessed resource and probes them in sequence ($f_{neg}() = \{0, 1\}$).

- HRL - A Heuristic based on Resource Load and the job at hand. Here, the Coordinator calculates the resource load, A^{ij}_{load}, relevant for a particular job i (Equation 4) and prioritises Resources ($j = \{1, N_{res}\}$) in the order of the least loaded Resource, min (A^{ij}_{load}), accessing them sequentially until the willing Resource is found.

$$A^{ij}_{load} = \frac{1}{C^j_{total}} \sum_{t=T^i_S}^{T^i_F} \sum_{n=1}^{C^j_{total}} C^j_n(t) \tag{4}$$

where C^j_{total} is the capacity of Resource j and is always 64 for the results presented. $C^j_n(t)$ is a Kronecker operator: equals 1 if node n of Resource j is booked/busy for time t and is set to 0 otherwise.

The first two algorithms were included mainly for the purpose of the model evaluation. LAF was chosen for its known performance [7] to be compared to HRL, which uses metadata described by the SLA to make an informed guess regarding the suitable Resource.

In Section 5 we present performance measurements for all four Coordinator types, using metrics described in this Section.

These algorithms have different $t_{dec}()$ values which we measured using our simulation algorithms performed in Java (Table 3). These, combined with negotiation over network values (not shown here) should give an adequate information as to the performance of each algorithm.

4.3 Resource

The roles of the Resource module are:

- to negotiate an agreement for job allocation according to the constraints specified in the SLA, based on the current information about its resource availability and scheduling heuristics.

- to form the maximum number of agreements by re-scheduling the job within the constrains specified in each SLA, i.e. without the need for renegotiation.

- to optimise resource utilisation.

- to provide a client with general information about the resource availability

Here we used a dummy scheduling mechanism in which each Resource knew which job requests can be accepted to ensure 100% utilisation. This way we can ensure that measurements obtained are only due to the Coordinator's algorithm.

5 Results and Discussion

We investigated the system with a variable number of CPU Resources per single Coordinator, $N_{res} =$

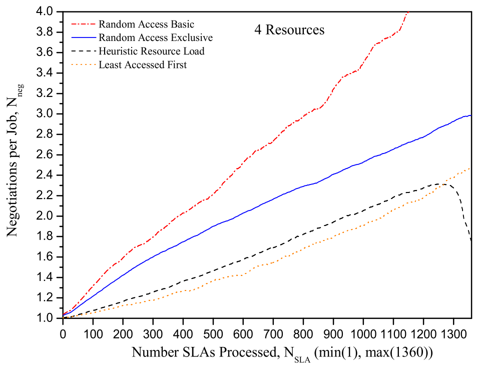

Figure 4: The performance of the Coordinator serving 4 Resources. The dependence of negotiation on the number of SLAs that has been agreed so far.

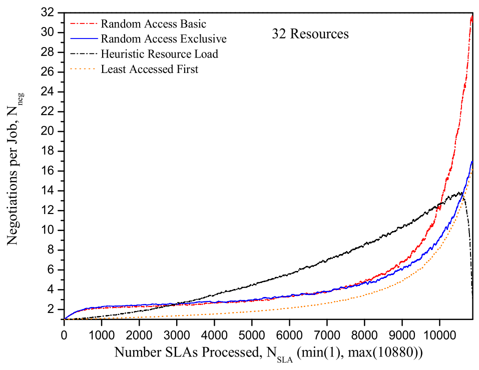

Figure 6: The performance of the Coordinator serving 32 Resources. The dependence of negotiation on the number of SLAs that has been agreed so far.

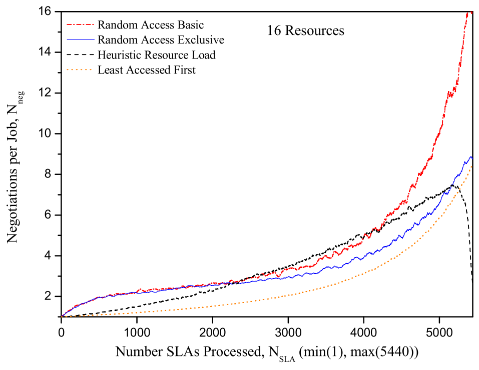

Figure 5: The performance of the Coordinator serving 16 Resources. The dependence of negotiation on the number of SLAs that has been agreed so far.

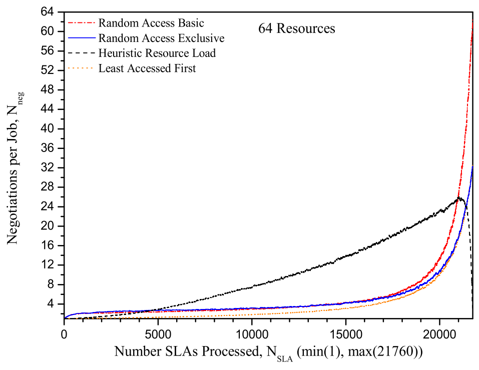

Figure 7: The performance of the Coordinator serving 64 Resources. The dependence of negotiation on the number of SLAs that has been agreed so far.

$\{2, 4, 8, 16, 32, 64\}$ and compared four different site selection algorithms that govern the Coordinator's decision. In Figures 4-7 we present the evaluation of the system in terms of the absolute number of negotiation attempts per successful SLA, averaged over 100-500 experiments, $\langle N_{neg}\rangle$. Each curve in Figures 4-7 was subsequently smoothed by adjacent averaging over 50 data points, which corresponds to $N_{SLA} = 50$ on these graphs. The curves show the dependence of $\langle N_{neg}\rangle$ on the number of successful SLAs or, generally speaking, the number of jobs processed (max $(N_{SLA}) = N_{job} \cdot N_{res}$ is equivalent to 100% load over N_{res} resources, f_{load} from Eq. 2).

Let us look first at the model with $N_{res} = 4$ (Figure 4). The best performing algorithm in general is LAF. Not far behind is HRL which starts to outperform the former in the region of high workload (higher $\sim$ 95%). The RAE and RAB algorithms require a higher number of negotiation attempts, on average, to achieve a successful SLA at any resource load. All four algorithms seem to perform with a fairly linear dependence on the resource load, apart from the HRL, which performs extremely well at very high resource loads. Similar, linear, performance was observed in [7] where the algorithm similar to LAF managed to allocate resources quicker than the random pick, when working with four resources.

As we increase the number of resources to 16 a new behaviour of the Coordinator's performance emerges (Figure 5). First of all, the average number of absolute negotiations for each successful SLA has increased, however we should also take into account the increased numbers of jobs processed and resources available (more on this later). Secondly, each algorithm (to a certain degree) deviates from its original linear behaviour and performs better, at least in the regions of low and medium resource load. The most interesting observation is, perhaps, the order in which algorithms rank in terms of performance. In the region of medium load, both random algorithms begin to outperform HRL, whereas LAF remains the leader, except in the region of very high load, giving way to HRL. With the further increase of $N_{res} = 32$ (Figure 6) the trend observed earlier becomes more profound. Both random algorithms outperform HRL, but this time the difference in performance and the scope is bigger. And finally, with the further increase of N_{res} to 64 resources (Figure 7) the same trend continues.

$t_{dec}(N_{res}, f_{neg})$	$N_{res} = 2$	$N_{res} = 4$	$N_{res} = 8$	$N_{res} = 16$	$N_{res} = 32$	$N_{res} = 64$
RAB, $t_{dec}(1)$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$
$t_{dec}(> 1)$	$0.07 \cdot 10^{-3}$	$0.07 \cdot 10^{-3}$	$0.07 \cdot 10^{-3}$	$0.07 \cdot 10^{-3}$	$0.07 \cdot 10^{-3}$	$0.07 \cdot 10^{-3}$
RAE, $t_{dec}(1)$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$
$t_{dec}(> 1)$	$0.2 \cdot 10^{-3}$	$0.2 \cdot 10^{-3}$	$0.3 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.6 \cdot 10^{-3}$	$0.9 \cdot 10^{-3}$
HRL^{10}, $t_{dec}(1)$	$0.9 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$4.1 \cdot 10^{-3}$	$10 \cdot 10^{-3}$	$31 \cdot 10^{-3}$	$103 \cdot 10^{-3}$
HRL^{100}, $t_{dec}(1)$	$4.2 \cdot 10^{-3}$	$8.6 \cdot 10^{-3}$	$19 \cdot 10^{-3}$	$37 \cdot 10^{-3}$	$84 \cdot 10^{-3}$	$215 \cdot 10^{-3}$
$t_{dec}(> 1)$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$
LAF, $t_{dec}(1)$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$1.3 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$21 \cdot 10^{-3}$	$84 \cdot 10^{-3}$
$t_{dec}(> 1)$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$	$0.4 \cdot 10^{-3}$

Table 3: Average performance figures, $t_{dec}(f_{neg})$ for different algorithms and N_{res}. For each prospective SLA at least one negotiation attempt is made, whilst t_{dec} before each attempt is different, so that $t_{dec}(1) \neq t_{dec}(\alpha)$ and $t_{dec}(\beta) = t_{dec}(\beta + 1)$, where $\alpha = \{2, N_{res}\}$, $\alpha \neq \beta = \{2, N_{res}\}$. For HRL two sets of data are provided: $S = 10$ - sum over ten elements, $S = 100$ - sum over hundred elements (in Equation 4), where $S \sim (T_F^i - T_S^i)$ for any given Job i. Times are given in milliseconds for a single decision made per successful SLA and are based on real times of Java code running on 3GHz processor.

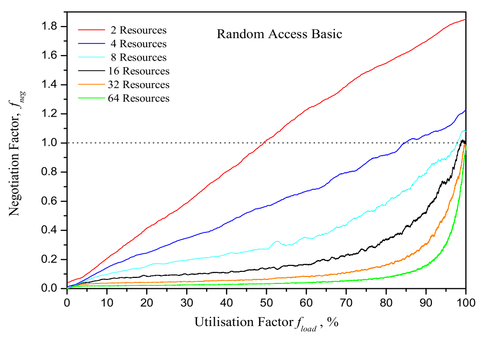

Figure 8: The dependence of the negotiation factor on the load factor, $f_{neg}(f_{load})$ for different number of Resources served by an RAB based Coordinator.

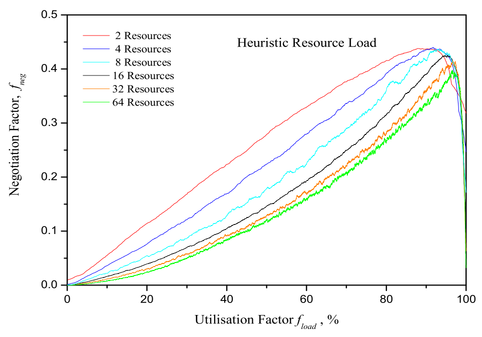

Figure 10: The dependence of the negotiation factor on the load factor, $f_{neg}(f_{load})$ for different number of Resources served by an RHL based Coordinator.

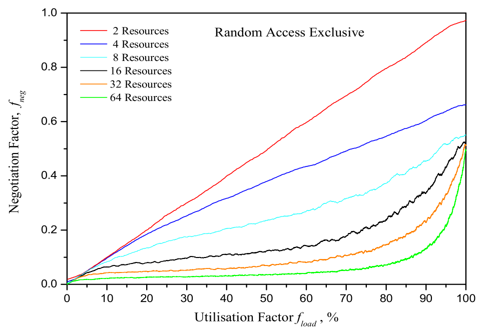

Figure 9: The dependence of the negotiation factor on the load factor, $f_{neg}(f_{load})$ for different number of Resources served by an RAE based Coordinator.

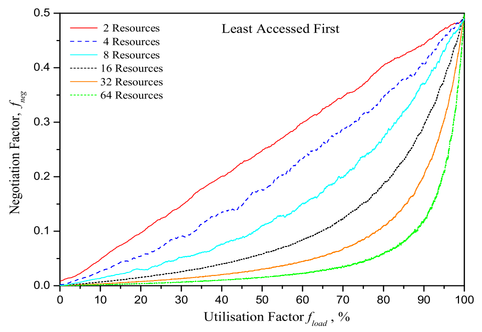

Figure 11: The dependence of the negotiation factor on the load factor, $f_{neg}(f_{load})$ for different number of Resources served by an LAF based Coordinator.

In all four cases (Figures 4-7) we observe an improvement in the performance of all four algorithms as the number of Resources served by a single Coordinator is increased, although the HRL algorithm remains fairly stable to the change.

In the region of large f_{load}, even the worst performing algorithm, RAB, performs reasonably well with high N_{res}, keeping the average number of negotiations close to the number of Resources available. A Coordinator serving a lower number of Resources makes more negotiation attempts per SLA than there are Resources even at lighter resource loads. An RAE based Coordinator achieves the rates of half of its possible maximum negotiations per SLA for high f_{load} values as compared to its own performance with smaller number of Resources. The variations in performance for HRL is least noticeable among all four. In addition, the peak on the graph (Figure 10) which represents the point of the worst performance shifts slightly toward the higher load as N_{res} increases. All algorithms approach a saturation in performance with increase in N_{res}. Table 3 outlines the running time for each algorithm presented here.

6 Conclusions and Future Work

We have built a model in which a job allocation mechanism may be explored by means of varying topology, negotiation processes, SLA specifications as well as various heuristics for Coordinator's site selection and Resource's scheduling algorithms. The model was tested using four simple algorithms. We found that the performance of the Coordinator may and usually does depend on the number of Resources served, precisely it improves with the increase of Resources in terms of negotiation attempts performed for each successful SLA. We showed that a simple heuristic based on data from SLA can provide more stable and, in places, better performance than that of other standard site selection algorithms. The important conclusion from this is that the HRL method behaves fundamentally differently from other methods that do not use SLA data.

Further work will investigate different SLA based scheduling algorithms, more accurately the evaluation of the Coordinator-Resource pair in different topologies and negotiation approaches. We also intend to widen the set of parameters outlined in the SLA and increase the overall complexity of the model, incorporating various time factors, identifying new metrics, *etc.* The new results and findings from the simulation are expected to feed into the enhanced requirements to SLA based job management, including extensions to the current WS-A specification.

Acknowledgement: This research has been funded by EPSRC (through grants GR/S67654/01 and GR/S67661/01) whose support we are pleased to acknowledge.

References

[1] Joy Hathaway, "Service Level Agreements: Keeping A Rein on Expectations", Winning The Networking Game, ACM (1995)

[2] "WS-Agreement", the White Paper to GGF, Draft 18 (14 May 2004)

[3] Jon MacLaren, Rizos Sakellariou, Krish T. Krishnakumar, Jon Garibaldi, Djamila Ouelhadj, "Towards Service Level Agreement Based Scheduling on the Grid", Workshop on Planning and Scheduling for Web and Grid Services (in conjunction with ICAPS-04; Whistler, British Columbia, Canada, 100 (3-7 Jun 2004)

[4] Karl Czajkowski *et al*, "SNAP: A Protocol for Negotiation of Service Level Agreements and Coordinating Resource Management in Distributed Systems", 8th Workshop on Job Scheduling Strategies for Parallel Processing.

[5] "Grid Resource Allocation Agreement Protocol", GGF Working Group, *https://forge.gridforum.org/projects/graap-wg*

[6] J. Padjett, M. Haji, K. Djemame, "SLA Management in a Service Oriented Architecture", ICCSA'05, LNCS 3483, 1282 (2005)

[7] C. Dumitrescu, Ian Foster, "GRUBER: A Grid Resource SLA Broker", University of Chicago, USA

[8] A. Dan, C. Dumitrescu, M. Ripeanu, "Connecting Client Objectives with Resource Capabilities: An Essential Component for Grid Service Management Infrastructure", ACM ICSOC'04, New York (2004)

[9] IBM, "WSLA Language Specification, ver. 1.0.", (2003)

[10] H. Ludwig, A. Dan, B. Kearney, "Cremona: An Architecture and Library for Creation and Monitoring WS-Agreements", ACM ICSOC'04, New York (2004)

[11] G. J. Henry, "A Fair Share Scheduler", University of Sydney, AT&T Bell Labs (1984)

[12] D. Kagklis, N. Liampotis, C. Tsakiris, "A Framework for Implicit and Explicit Service Activation Based on Service Level Specification", SAC'04, 363 (14-17 Mar 2004)

[13] V. K. Naik, S. Sivasubramanian, S. Krishnan, "Adaptive Resource Sharing in a Web Service Environment", Middleware '04 LNCS 3231, 311 (2004)

[14] Yun Fu, Amin Vahdat, "SLA Based Distributed Resource Allocation for Streaming Hosting Systems", *http://issg.cs.duke.edu*

[15] Mladen Vouk, Yannis Viniotis, "Satisfaction of Service-Level Agreements", North Carolina State University, web presentation *http://renoir.csc.ncsu.edu/Faculty/Vouk/Papers/Slides/SLA_CACC_Nov01.pdf* (Nov 2001)

[16] Li Zhang, Danilo Ardagna, "SLA Based Profit Optimization in Autonomic Computing Systems", ICSOC'04, 173 (15-19 Nov 2004)

[17] Zhen Liu, Mark S. Squillante, Joel L. Wolf, "On maximizing Service Level Agreement Profits", Technical Paper, IBM T.J. Watson Research Center, NY 10598, USA

[18] Web based auction, *http://www.ebay.co.uk*