Job Scheduling on the Grid: Towards SLA-Based Scheduling

Rizos SAKELLARIOU [a,1] and Viktor YARMOLENKO [a]

[a] *School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom.*

Abstract. This paper argues for the need to provide more flexibility in the level of service offered by Grid-enable high-performance, parallel, supercomputing resources. It is envisaged that such need could be satisfied by making separate Service Level Agreements (SLAs) between the resource owner and the user who wants to submit and run a job on these resources. A number of issues related to the materialization of this vision are highlighted in the paper.

Keywords. Grid Computing, Job Scheduling, Service Level Agreement

1. Introduction

Traditionally, the scheduling mechanisms used for the enactment of jobs on parallel supercomputer resources have been queue-based. Such mechanisms are essentially offering only one level of service, which can be summarized simply as 'run the job when it gets to the head of the queue' (even though, in some cases, some jobs that are behind in the priority queue may be assigned to unutilized processors, to ensure that there are no idle processors, using a technique known as backfilling [1,2]).

The emergence of Grid Computing [3] has created new opportunities to support compute and/or data intensive scientific applications, which, among other, may have large computational resource requirements. However, at the same time, Grid computing is built upon the notion of *virtual organizations* [4]. This is a group of individuals and/or institutions who collaborate towards the solution of a particular problem through a set of resource sharing rules. The key implication of this *coordinated resource sharing* [4] is that the collaboration of the members of a Virtual Organization may span across different administrative domains. In the context of scheduling independent jobs of a large, parallelizable application that makes use of the Grid, such a collaboration, across multiple administrative domains, would also require some coordination across the different schedulers of the individual resources employed. Lack of coordination may annul all the benefits from parallelism that one might expect when independent jobs are running, in parallel, onto different resources.

To illustrate this problem, consider an application that can be represented by a Directed Acyclic Graph (DAG). Different nodes of the graph represent individual jobs

[1]Corresponding Author: Rizos Sakellariou, School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom; E-mail: rizos.sakellariou@manchester.ac.uk.

with computational requirements, whereas edges represent communication of data. In the context of Grid computing, DAGs can be used to represent many applications that belong to the important family of applications collectively known as *scientific workflows* [5,6,7,8,9,10,11]. In order to exploit the task parallelism available in the DAG (by executing independent nodes in parallel), let us assume that two different nodes are assigned to different resources, each resource belonging to a different administrative domain and using its own scheduler with its own queueing system. Since each job will start execution when it reaches the head of the local queue (of the corresponding resource), it is possible (because of different waiting times) that one job may finish execution before the other job has even started. From a global point of view, this is equivalent to executing the two jobs sequentially. In other words, there are no performance benefits due to the exploitation of parallelism because of the different behaviour of the queue-based schedulers of the local resources, which have acted without coordination and contrary to user expectations.

Advance Reservation of resources has been suggested as a means to guarantee that tasks will run onto a resource when the user expects them to run [12,13]. Essentially, advance reservation specifies a precise time that jobs may start running. This allows the user to request resources from systems with different schedulers for a specific *time interval* (e.g., start time, finish time), thereby obtaining a sufficient number of resources for the time s(he) may need. Advance reservation has already received significant attention and has been considered an important requirement for future Grid resource management systems [14]. There has been already significant progress on supporting it by several projects and schedulers, such as the Load Sharing Facility platform (LSF) [15], Maui [16], COSY [17], and EASY [18,19].

However, advance reservation is again an extreme level of service since it specifies a precise time when jobs can be made to run. This may cause several problems to the resource owner, and there is some scepticism in the community, especially with respect to the degree to which advance reservations contribute to improving the overall performance of a scheduler [20]. For example, when an advance reservation is made, the scheduler must place jobs around this fixed job. Typically, this is done using backfilling [2], which increases utilisation by searching the work queues for small jobs to plug the gaps. In practice, this rarely works perfectly, and so the scheduler must either leave the reserved processing elements empty for a time, or suspend or checkpoint active jobs near to the time of the reservation. These processes are not instantaneous; e.g., checkpointing a 64 processor Unified Weather Model job on an O3800 takes about 12 minutes, despite a small total memory footprint of 3Gb; checkpointing 256 processor jobs can exceed one hour. Suspension is faster, but can adversely affect the performance of the incoming job, due to the cost of swapping out memory used by the suspended job when it is required by the incoming job. Either way, there are gaps in the schedule, i.e., CPU time which is not processing users' work. As utilisation often represents income for the service owner, there is a tendency to offset the cost of the unused time by charging for advance reservation jobs at a considerably higher tariff. While it is possible to set tariffs high enough to compensate, this brute-force solution is inefficient in terms of resources, and undesirable for both users, who pay higher prices, and for resource owners, who must charge uncompetitive prices and watch utilization fall.

In recent years, there has been work aiming to explore the space between the two aforementioned extreme levels of service, namely, 'run this job whenever it gets to the head of the queue', or 'run this job at this precise time'. The main idea is to provide different levels of service by forging agreements between the different parties (user, resource owner, etc). Such agreements are agreed on the basis of different constraints expressed by the user and/or the resource owner and essentially specify a desired (and agreed) level of service. The use of *Service Level Agreements* (SLAs) gives rise to a fundamentally new approach for job scheduling on the Grid.

This paper provides an overview of the issues surrounding the design of a fundamentally new infrastructure for job scheduling on the Grid, which is based on the notion of Service Level Agreements (SLAs), based on work carried out as part of a recently completed project [21]. In the most general form, such SLAs are negotiated between a client (user, superscheduler, or resource broker) and a provider (the owner of a resource with its own scheduler), may contain information such as acceptable job start and end times, and may be re-negotiated during runtime.

Some background and an overview of relevant work on SLAs as well as the ways that they can be used on the Grid is given in Section 2. Section 3 describes the key features of the envisaged architecture for job scheduling using SLAs on the Grid. Section 4 describes the key issues and challenges that need to be addressed for the materialization of such an architecture. Sections 5 and 6 elaborate on approaches to address some of these issues, namely the description of the terms in an SLA as well as the design of heuristics for SLA scheduling, building on earlier work presented in [22,23,24]. Finally, Section 7 concludes the paper.

2. Background and Related Work

An SLA can be described as a legally binding contract between the parties involved. The agreement relates to a transaction for the provision of a service; as a result, the parties of an SLA can be distinguished between providers and consumers of a service. The terms of the SLA describe the expected level of service within which the service will be provided; these have been agreed between service providers and service consumers.

It has been mentioned that forms of SLAs were in operation since the 1960s, "when they were used as a method for buying minutes of computer machine time" [25]. In more recent years, SLAs became more widespread as a means to make agreements when outsourcing IT functions [26], or when providing network services [27,28,29]. Most of these agreements were paper-based and were drawn after some form of negotiation between appropriate persons.

The shifting emphasis of the Grid towards a service-oriented paradigm — as well as trends in application service delivery to move away from tightly coupled systems towards structures of loosely coupled, dynamically bound systems [30] — led to the adoption of Service Level Agreements as a standard concept by which work on the Grid can be allocated to resources and enable coordinated resource management. In the context of Grid and Web services, the current understanding of the community is that such an SLA is essentially an electronic contract, which is expected to be negotiated fully automatically (i.e., without any human intervention) by different processes and, as such, much be machine readable and understandable.

As a result, there has been a significant amount of research, in recent years, on various topics related to SLAs. Issues related to their overall incorporation into grid architectures have been discussed in [31,32,33,34,35]. Issues related to the specification of the SLAs have been considered in [36,37,38,39]. Issues specifically related to (and motivated from) the usage of SLAs for resource management on the Grid have been considered in [34,40,41,42,43,44,45]. Of particular importance has been the work on the negotiation of SLAs. Since the early influential work on the Service Negotiation and Acquisition Protocol (SNAP) [41], further work has argued for the need to take into account the principles of contract law when negotiating SLAs in order to form legally binding agreements [46], even considering how the consequences from the use of SLAs may need to be taken into account by guidelines on electronic contracts [47]. Other work has examined issues related to trust and security [48,49], or targeted more business oriented case studies [50]. Later work has also drawn upon relevant research carried out particularly in the context of agents [51,52], where several techniques have been used to model negotiation, ranging from heuristics to game theoretic and argumentation-based approaches [53]. Finally, a significant area of research relates to the economic aspects associated with the usage of SLAs for service provision (e.g., charges for successful service provision, penalties for failure, etc.); relevant work has been presented in [54,55, 56,57,58,59].

Work within the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group of the Open Grid Forum [60] (and earlier within its predecessor, the Global Grid Forum) has led to the development of *WS-Agreement (WS-A)* [61], a specification for a simple generic language and protocol to establish agreements between two parties. Each of the two parties can be either an initiator of or a responder to the agreement. The agreement structure is composed of several distinct parts, namely Name, Context and Terms of Agreement. The latter is also divided in service description terms and guarantee terms. Service descriptions terms mainly describe the functionality to be delivered under the agreement. The guarantee terms define the assurance on service quality for each item mentioned in the service description terms section of the WS-A. In the specific context of job submission, which is the focus of this paper, such assurances may be defined as a parameter (constant) or bounds (min/max) on the availability of part or the whole of the resource. In WS-A, such assurances are referred to as service level objectives (SLOs); in a domain specific to computation services provision, they are usually expressed as values (e.g., SLO: CPUcount = 8). Each SLO may refer to one or more business values, called a business value list (BVL). This list expresses different value aspects of a specific SLO. The other two types of guarantee terms are Qualifying Conditions and Importance, which have a similar function to SLO and BVL, respectively.

3. An Architecture for Job Scheduling on the Grid Using Service Level Agreements

The key vision of this paper is that jobs, submitted for execution to high-performance computing resources, are associated with an SLA. This SLA is negotiated between a client (e.g., a user or a resource broker) and a provider (the owner of a resource with its own local scheduler) and contains information about the level of service agreed between the two parties, such as acceptable job start and end times.

An overall architectural view, materializing this vision, is presented in Figure 1. There are three key 'players' underpinning this materialization. *Users* negotiate and agree an SLA with a resource broker (or superscheduler, or coordinator). *Brokers* nego-

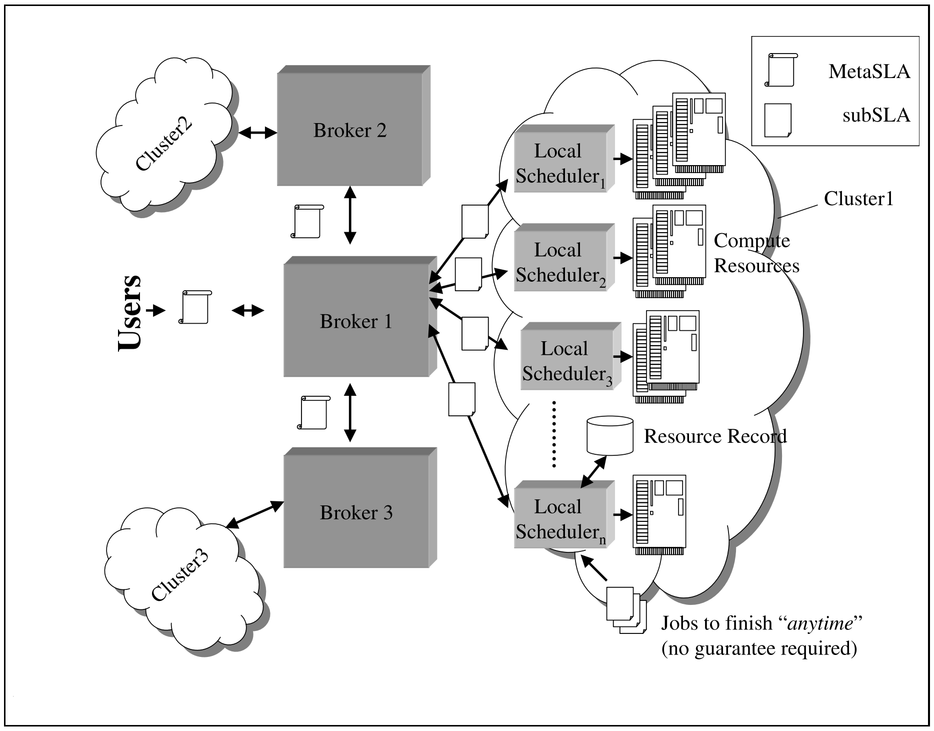

Figure 1. An overview of an architecture for SLA-based job scheduling.

tiate and agree an SLA with users; these SLAs may be mapped to one or more SLAs, which are negotiated and agreed with local resources and their schedulers. Finally, *local schedulers* need to schedule the work that is associated with an SLA which they agreed to (the constraints associated with such an SLA, agreed by a resource, may be stored locally in the resource, in some kind of a resource record). It is also noted that a single SLA agreed between a user and a broker may 'translate' to multiple SLAs between the broker and different local resources to serve the user's request (for example, this could be the case when the SLA between a user and a broker refers to a workflow application with several tasks that are executed on different resources. In such case, the user may want to set constraints for the workflow as a whole and the broker may have to translate it to specific SLAs for individual tasks, following an approach similar to the one described in [62]); to indicate the possible differences between these two types of SLA, the terms *meta-SLA* and *sub-SLA* are used. Furthermore, as indicated in the figure, this SLA-based view for job submission, may still allow the submission of jobs that are not not associated with an SLA; however, no guarantees about their completion time would be offered in this case.

It should be stressed also that the primary objective of the architectural view presented in Figure 1 is to illustrate the fundamentally different interactions that may arise as a result of the use of SLAs to make an agreement on the level of service expected when jobs submitted. It is beyond the scope of this paper to argue for or against a particular type of architecture. For example, one may assume that a broker is associated with a single cluster (as in the figure), within the same administrative domain, or with many, across different administrative domains. Similarly, the broker may be more tightly connected to the local schedulers (in which case, the broker may exercise more control on the local schedulers and no SLA between the broker and the local schedulers may be necessary), or no SLAs may be agreed between different brokers. However, besides the pros and cons of different architectural choices, there are several common challenges that need to be addressed; this is the main focus in the remainder of this paper.

4. Issues and Challenges with SLA-Based Job Scheduling

This section groups some of the key challenges that need to be addressed in order to materialize an SLA-based vision for job submission and scheduling. These challenges could be summarized as follows:

SLA vocabulary: The vision of SLA based scheduling assumes that the SLAs themselves are machine readable and understandable. This implies that any agreements, between the parties concerned, for a particular level of service need to be expressed in a commonly understood (and legally binding) language. There has been early work on generic languages for SLA description [38,39], but none related to the particular problem of the requirements associated with job submission and execution (on possibly high-performance computing resources). Some of the issues that may arise in such cases, and some of the trade-offs in the description of the vocabulary are discussed in the next section.

Negotiation: It is envisaged that SLAs may be negotiated between machines and users or only between machines. In this negotiation some commonly agreed protocol needs to be followed. This protocol needs to take into account both the nature of the distributed systems and networks which are used for the negotiation (for example, what if an offer from one party is not received by the other party due to a network failure), and should abide by appropriate legal requirements (for example, should the receipt of every proposal for an agreement be acknowledged or not?). In addition, during negotiation, machines should be able to reason about whether an offer is acceptable and possibly they should be able to make counter-offers. As already mentioned in Section 2 there has been a significant amount of work on these topics. However, the relevant challenges on negotiation are more of a generic nature rather than specific to the problem of using SLAs for job scheduling.

Scheduling: Given that, currently, scheduling of jobs on high-performance compute resources is mostly based on priority queues (with the possible addition of backfilling techniques [1,2]), the use of SLAs would require the development of a new set of algorithms for efficient scheduling, which would be based on satisfying the terms agreed in the SLA. The existence of efficient scheduling algorithms would be of paramount importance to estimate capacity and reason on the possible acceptance (by a local resource) of a new request to make an SLA. Some approaches related to the development of such algorithms are discussed further later on.

Constitutional Aspects: In the context of any SLA based provision, sooner or later, the need for dispute resolution may arise. In addition, users may also be interested in the reliability of specific brokers; for example, how likely (or unlikely) is that a broker will honour an SLA (even if breaking the SLA would require the broker to pay a penalty). This issue of modelling reputation may also be related to the approaches followed for pricing and/or penalties when agreeing SLAs (for example: is there a flat charge for the usage of resources? do the fees vary depending on particular circumstances?). Such issues, might become more important as the envisaged usage of SLA-based job scheduling grows. However, since relevant work is in its infancy, simple assumptions, without neglecting the challenges that might arise later in this respect, could be made for a start.

A more detailed overview of the issues and trade-offs involved with particular choices with respect to the first and the third challenge above is given in the following two sections.

5. Issues on the Expressiveness of SLAs

Experience from trying to incorporate SLAs in distributed grid computing has indicated that this task is not as easy as defining a dictionary of terms, which are then placed in some kind of an XML derived structure. For example, work within the Open Grid Forum (OGF) to define an SLA specification, has lasted for years and there have been several debates about what an SLA should contain and/or how it should look like. For instance, one of the difficulties that had to be addressed related to how issues related to the SLA document description would be decoupled from issues related to the negotiation protocols.

It must be noted here, that an efficient SLA related environment, with guarantees for a certain level of fault tolerance, may often include renegotiation [27,31,41] of the whole or part of an SLA. In the context of SLA-based job scheduling, simulation studies [44] have indicated that, at high resource load, a significant proportion of the agreed SLAs may have to be renegotiated in order to avoid failure of an SLA. Such renegotiation may have an overhead and may require the user's participation at one or more stages between the time when the initial agreement was made and when the work, specified in the SLA, is completed. It would clearly undermine the overall vision of Grid, as an environment based on the service-oriented paradigm and ultimately driven by principles similar to those of autonomic systems [63,64], to require a possibly repeated user involvement every time renegotiation is in order. One would expect some abilities of self-management in this respect.

The trade-off (and the dilemma), here, is whether a much simpler SLA description (which, however, may cause more often the need for renegotiation) is more (or less) desirable than a more complex SLA description, which, however, enables the system to deal more effectively (and possibly transparently from the user) with the need for renegotiation. In other words, simplicity in the SLA description itself at the expense of renegotiation overhead, or more efficient scheduling of the SLAs [44,24] at the expense of more complex SLA descriptions? The idea first presented in [22,44] attempts to answer this question by making SLAs (infinitely) more *expressive*. Such increased expressiveness in SLAs is achieved by describing SLA terms as arbitrary *analytical functions*, as opposed to the traditional approach of specifying SLAs [61] that uses constant values or ranges for SLA terms. The authors in [22,44] have argued that the encapsulation of SLA terms in analytical functions can potentially reduce the need for renegotiation of a service and provide the extra flexibility needed for a wider spectrum of resource management decisions. The motivation, an example and some issues related to this approach to make an SLA more expressive will be described next.

In most cases of SLA-based resource management applications [33,65], the set of guarantee terms is rigidly defined. Within such an SLA, it is known in advance, for example, what the exact financial gain will be when an SLA is audited. There could be reasons to believe that this arrangement is not flexible and carries too little information for the provider bound by this SLA to perform its tasks successfully without the need for renegotiation. The expressive capacity of an agreement can be improved by adding more terms, but there is a practical limit on the SLA size, whilst most of the terms could easily be described analytically. The latter description can provide an infinitely large set of term configurations in SLA. Analytical functions can describe complex relationships between SLA terms that could potentially be used by autonomic applications in providing qualitatively new levels of service, described in an SLA with a higher degree of expressiveness. Such an SLA cover an infinitely large number of (agreed and acceptable) outcomes that belong to the continuum defined by the system of analytical functions.

To illustrate all this, we use an example from [23]; this relates to the reservation of resources for a computational job where using a different number of parallel processors to run the job would still result in a successful SLA from the user point of view. Naturally, the running time of such a (parallel) job would vary, depending on the parallelism assigned to it by the scheduler (that is, the number of processors assigned). The use of analytical functions means that the scheduler can now schedule and repeatedly reschedule this job in order to meet the objectives of its associated SLA without the need for renegotiation. Thus, the SLA terms relating to the size of the job (with respect to execution time needed and how this may relate to CPUs used) may look like:

- Maximum number of CPU Nodes that can be used, N_{CPU}^{max}
- CPU Nodes reserved, $N_{CPU} = \{2, 3, 4, ..., N_{CPU}^{max}\}$
- Reserved time for job execution, $t_D = \frac{t_D^1}{N_{CPU}}$

where t_D^1 is the projected time for the job to complete if it runs on a single CPU Node; and N_{CPU}^{max} is limited by the capacity of the resource. It can be seen that the reserved time for job execution is described as a function of the number of CPU nodes reserved. If a function was not used, then the SLA would have to include a single value for each of N_{CPU} and t_D; any attempt to deviate from these values would trigger the renegotiation of the SLA.

Of course, the relation between t_D and N_{CPU} is, in reality, more complex than what is described in the function used above. However, the point to make is that, even in this simple case, where SLA terms describe a limited number of configurations (N_{CPU}^{max} to be precise), the number of possible renegotiations needed would be prohibitively high, had all this been described in the traditional way as just a list of terms. Simulation shows that, even in such a simple case, the negotiation overhead per job decreases dramatically when the description in the SLA makes use of functions to link terms. It also suggests that by describing more SLA terms as functions, it is possible to achieve maximum utilisation with minimum negotiation. Moreover, novel scheduling algorithms may now be deployed, something that was not possible before, precisely because of the renegotiation cost. In [23], more examples of using an SLA with variable network bandwidth or resource load are presented; including some consideration of pricing terms.

The main argument that can be made against this approach (of describing SLAs in a more expressive manner) is the perceived complexity of the agreement. Indeed, for the SLA terms that are defined by very complex functions that in turn depend on other SLA terms, the agreement itself becomes a system of equations that may suffer from discontinuities, may have holes in solutions or may have no solution at all. Such representations of SLA terms could potentially put a very heavy strain on clients and providers, who would have to verify the SLA, check for contradictions, run away arguments and other unpleasantries. However, many of the problems would be reasonably easy to detect already and the complexity of the SLA could grow gradually with the readiness of environment, without any changes to SLA itself. Furthermore, a conservative approach could also be applied when negotiating SLAs. As in real life, if one party cannot understand all the terms in a proposed agreement, there is no obligation to agree.

Another important point made in [23], which might be overlooked easily, relates to the so-called set of *universal* terms. These are terms whose value is not known at the time an SLA is formed; they can be combined with analytical functions to make the SLAs more expressive. The most obvious argument in favour of the universal terms relates to the ability to enable agreements on terms (to form an SLA), the status of which may not be known at the time of making the agreement; yet, keeping the SLA in a concise form. To give an example, the actual amount of network traffic that a job will incur may not be known at the time an SLA is formed. Still, the existence of a universal term for network traffic could be used, for instance, in the earlier example as part of the analytical function describing the reserved time for job execution.

The use of universal terms makes it possible to create more speculative agreements, which encompass more possible outcome scenarios and uncertainties in a single concise SLA document, reducing further the need to renegotiate the agreement in the case of a failure (that has happened or is anticipated). To give an example, one such *universal* term, wall-clock time, was used in [24] to agree on the price for the service. The users were prepared to pay a higher price if the job was executed sooner rather than later. Hence, the price term, agreed as a function of the wall clock time (and other terms), enabled the SLA to cover an infinite number of possible outcomes with respect to the time constraints. This, in turn, enabled the scheduler to make more intelligent decisions in maximising profit for its owner, since the total revenue depending on different schedules could be calculated and taken into account. Such analysis would not be practical with traditional SLAs (that is, SLAs that do not make use of functions as described here to allow for more expressiveness), because the scheduler would have to renegotiate the pricing with hundreds of users.

6. A Review of Issues on Scheduling SLAs

Despite the growing interest in Service Level Agreements, there has been surprisingly little research published on the topic of using the information contained in the (agreed) SLAs for planning and scheduling purposes (with the objective of satisfying the requirements of the SLAs successfully). In the context of SLA-based job scheduling, the very idea of using SLAs is to alter the approach used to perform job scheduling. This means that local schedulers of high-performance computing resources must now take into account not only functional properties of the submitted job (such as parallelism and execution time) but also non-functional requirements expressed as terms and constraints in SLA. Even though the existences of various constraints that need to be satisfied may point to a constraint satisfaction problem, standard methods based on various forms of (exhaustive) searching would not be practical in a grid environment where a lage number of SLAs may have to be negotiated and scheduled quickly. Simple scheduling heuristics could provide efficient scheduling solutions with negligible time overheads for a large set of SLAs; a number of different scheduling heuristics was investigated in [24].

The principle behind these heuristics is based on how SLAs would be prioritised. Scheduling, then, falls to a simple routine of picking jobs (in order of priority) from the prioritised list and fitting them onto the resource, without rescheduling previous jobs already allocated – a single iteration packing process.

The priority value, H, is computed using a function, whose generic form is as follows:

$$H = \min\left(h_1 + w \cdot h_2\right) \tag{1}$$

$$H = \max\left(h_1 + w \cdot h_2\right) \tag{2}$$

where h_1,h_2 is one of:

- Earliest job start time, T_S
- Latest job finish time, T_F
- Reserved time for job execution, t_D
- Number of CPU Nodes required, N_{CPU}
- Job size, measured in CPU-hours, $A = N_{CPU} \times t_D$
- Job laxity, defined as $t_L = T_F - (T_S + t_D)$

and w is a weighting coefficient that can be both positive and negative. Obviously, for each heuristic, h_1 and h_2 are different, so the total number of heuristics that can be created combining all the terms above with each other is 15. By sweeping across values of w in equations 1 and 2 the best effort configuration can be found for each heuristic.

To test these SLA-aware scheduling heuristics the following scenario and SLA template were used. In a simple model, which consists of a *Client*, a *Provider* and an *Agreement* (SLA) between the two, the requests from clients (bound by an SLA) were scheduled by the local resource scheduler; the latter used different heuristics for the local scheduling of these requests. The SLA in this case consisted of the following five terms: (i) Earliest job start time, T_S; (ii) Latest job finish time, T_F; (iii) Reserved time for job execution, t_D; (iv) Number of CPU Nodes required, N_{CPU}; and (v) Final price agreed, V_{TOT}. It was assumed that two different pricing policies were used to calculate the income for the Provider:

- *flat rate*: The charge for each SLA is the same, regardless of duration of the job, number of CPUs used, etc.
- *pay as you go*: The charge is proportional to the actual usage of the resource (that is, the product of the time and the number of CPUs used).

heuristic	SLA%	CPU%	w
$\min(T_F + wN_{CPU})$	97.0%	84.8%	0.48
$\min(T_F + wA)$	96.8%	82.8%	0.06
$\min(T_S + wA)$	96.1%	83.0%	0.28
$\min(T_F + wt_D)$	95.4%	88.0%	-0.025
$\min(T_S + wt_D)$	95.4%	87.0%	10.0
$\min(T_F + wt_L)$	95.3%	86.3%	0.015
$\min(T_S + wt_L)$	95.3%	86.0%	1.15
$\min(T_F + wT_S)$	95.3%	85.5%	-0.05
$\min(T_S + wN_{CPU})$	92.7%	77.5%	4.5

Table 1. Evaluation of the performance of different SLA scheduling heuristics using a flat rate pricing policy.

heuristic	SLA%	CPU%	w
$\min(T_F + wt_D)$	92.0%	94.0%	-6.63
$\min(T_S + wt_L)$	92.2%	93.9%	0.4
$\min(T_F + wT_S)$	91.9%	93.9%	2.0
$\min(T_F + wt_L)$	92.1%	93.7%	-0.7
$\min(T_S + wt_D)$	91.7%	93.7%	3.2
$\min(T_S + wN_{CPU})$	85.0%	93.3%	-0.63
$\min(T_F + wA)$	89.7%	93.2%	-0.3
$\min(T_S + wA)$	89.0%	92.8%	0.0
$\min(T_F + wN_{CPU})$	90.5%	89.3%	-3.0

Table 2. Evaluation of the performance of different SLA scheduling heuristics using a flat rate pricing policy.

Nine of the fifteen heuristics that could be created following the approach described above, were evaluated. The choice of these nine heuristics was based on experience from results in [24]. Thus, the nine heuristics refer to all combinations that include at least one of the terms T_F and T_S. For the evaluation, we used a simple (synthetic) SLA workload model whereby, for each SLA workload set generated, a scheduling solution existed that resulted in a 100% resource utilisation for the SLAs in the set. For each of the two pricing policies, 100 SLA workload sets were generated for a single experiment. The results then were averaged over 100 experiments. On average, each SLA workload consisted of around 380 SLAs. The latter were supposed to be scheduled on a homogeneous parallel resource of 48 CPU nodes, which was available for 400 hours, creating thus a reference frame of 48×400 in size. Each of the different SLA workloads was scheduled on the resource of the same size using all nine different heuristics.

The averaged results are shown in Table 1 (for the 'flat rate' pricing policy) and in Table 2 (for the 'pay as you go' pricing policy). The results show, for each heuristic considered (first column): the average percentage of successfully scheduled SLAs out of all the SLAs that could be scheduled in principle (SLA satisfiability – column 2); the average percentage of hours the CPU was busy out of the total number of hours available (CPU utilization — column 3); and the value of w that was chosen to maximize income for each pricing policy (column 4). In each table, the results are sorted starting from the heuristic that provides on average the highest income (clearly, in the case of the 'flat rate' pricing policy the income is proportional to SLA satisfiability, while in the case of the 'pay as you go' pricing policy the income is proportional to CPU utilization).

Our experiments revealed several interesting points:

- Depending on the pricing policy used it was possibly to achieve either high SLA satisfiability (over 95% in the case of the 'flat rate' pricing policy) or high CPU utilization (over 89% in the case of the 'pay as you go' pricing policy). It is noted that these are average values; in individual cases (especially on a small-scale reference frame), these percentages could be even higher. These are very encouraging results considering the SLA scheduling problem, especially if we take into account that scheduling was performed in a fraction of a second.
- The performance of the heuristics appears to be related to the pricing policy used. Thus, heuristics that achieve the highest percentage of SLA satisfiability using the 'flat rate' pricing policy do not seem to achieve the highest percentage of CPU utilization using the 'pay as you go' pricing policy and *vice versa*. This observation is consistent with the results in [24] where it was found that the choice of the pricing policy can change dramatically the effect of different heuristics. On average, the heuristic based on T_F and T_D appears to perform best in both cases.
- Optimizing for CPU utilization (as in Table 2) appears to provide more robust results in the sense that the level of SLA satisfiability is not affected as much as CPU utilization is affected when the objective is to optimize for SLA satisfiability (as in Table 1).
- When the job laxity reaches $\approx 0.5 \cdot 10^2$ of t_D the time constraints of the job appear to no longer affect the performance of any of the considered scheduling heuristics.
- On average, when the ratio of the average value of N_{CPU} to the size of the resource was, roughly, 1:8 most heuristics seemed to result in the highest utilisation.
- The performance of some heuristics was very sensitive to the workload. For example heuristics based on N_{CPU} and A change in a very nontrivial way with the change of the distribution of CPU requests.

It must be said here that this was a limited study. Although it indicated that heuristics could be used to provide good solutions to the scheduling problems, additional studies, with varying workloads are highly desirable.

7. Conclusion

This paper argued for the need of a fundamental approach for job scheduling on (parallel) high-performance computing resources, based on service level agreements. This approach appears to become more pertinent with developments in the context of Grids, service-oriented architectures and autonomic computing. The paper highlighted a number of issues that need to be addressed in order to materialise such a novel approach. Some work aiming to address some of these aspects, in particular related to the description of the SLA terms and their use in scheduling, was briefly presented. The authors' view is that SLA based approaches for resource provision are highly promising; however, there is a need for further advances and research in the challenges indicated before such approaches become common-place.

Acknowledgements

The research presented in this paper has been funded by EPSRC (Grant Reference: GR/S67654/01); their support is gratefully acknowledged. Support from the CoreGRID Network of Excellence is also acknowledged.

References

[1] Dror G. Feitelson and Ahuva M. Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling. In *Proceedings of the 12th International Parallel Processing Symposium*, pages 542-546, April 1998.

[2] David A. Lifka. The ANL/IBM SP Scheduling System. *Proceedings of the IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, volume 949, pp. 295-303, 1995.

[3] Ian Foster and Carl Kesselman (eds). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[4] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), pp. 200-222, 2001.

[5] The Workshop on Workflows in Support of Large-Scale Science (WORKS06), in conjunction with HPDC2006, Paris, 2006. http://www.isi.edu/works06

[6] The 2nd Workshop on Workflows in Support of Large-Scale Science (WORKS07), in conjunction with HPDC2007, Monterey Bay, California, 2007. http://www.isi.edu/works07

[7] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, Anirban Mandal, and Ken Kennedy. Task scheduling strategies for workflow-based applications in grids. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pp. 759-767, 2005.

[8] Anirban Mandal, Ken Kennedy, Charles Koelbel, Gabriel Marin, John Mellor-Crummey, Bo Liu, and Lennart Johnsson. Scheduling Strategies for Mapping Application Workflows onto the Grid. In *IEEE International Symposium on High Performance Distributed Computing (HPDC 2005)*, 2005.

[9] Arun Ramakrishnan, Gurmeet Singh, Henan Zhao, Ewa Deelman, Rizos Sakellariou, Karan Vahi, Kent Blackburn, David Meyers, and Michael Samidi. Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007, pp. 401-409.

[10] Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Schields. *Workflows for e-Science. Scientific Workflows for Grids*. Springer, 2007.

[11] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. In *SIGMOD Record*, volume 34(3), September 2005.

[12] Jon MacLaren. Advance Reservations: State of the Art. In Global Grid Forum 9 (GGF9), Scheduling and Resource Management Workshop, Chicago, USA, October 2003.

[13] Warren Smith, Ian Foster, and Valerie Taylor. Scheduling with Advanced Reservations. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, pages 127-132, May 2000.

[14] Uwe Schwiegelshohn, Philipp Wieder, and Ramin Yahyapour. Resource Management for Future Generation Grids. In *Future Generation Grids*, Vladimir Getov, Domenico Laforenza, Alexander Reinefeld (Eds.), Springer, CoreGrid Series, 2005.

[15] Load Sharing Facility platform. *http://www.platform.com/products/LSF/*.

[16] David B. Jackson, Quinn Snell, and Mark J. Clement. Core Algorithms of the Maui Scheduler. In *Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Lecture Notes in Computer Science, volume 2221, pp. 87-102, 2001.

[17] Junwei Cao and Falk Zimmermann. Queue Scheduling and Advance Reservation with COSY. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS'04)*, Santa Fe, USA, April 2004.

[18] David A. Lifka, Mark W. Henderson, and Karen Rayl. Users Guide to the Argonne SP Scheduling System. Technique Report ANL/MCS-TM-201, Argonne National Laboratory, May 1995.

[19] Joseph Skovira, Waiman Chan, Honbo Zhou, and David A. Lifka. The EASY – LoadLeveler API Project. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, volume 1162, pp. 41-47, 1996.

[20] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel Job Scheduling — A Status Report. In *Revised Selected Papers from the 10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004)*, Lecture Notes in Computer Science, volume 3277, pp. 1-16, 2004.

[21] Service Level Agreement Based Scheduling Heuristics. EPSRC funded project, GR/S67654/01.

[22] Rizos Sakellariou and Viktor Yarmolenko. On the Flexibility of WS-Agreement for Job Submission. In *Proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC'05)*, Grenoble, France, November 28 - December 02, 2005, ACM International Conference Proceedings Series, vol. 117, 2005.

[23] Viktor Yarmolenko and Rizos Sakellariou. Towards Increased Expressiveness in Service Level Agreements. *Concurrency and Computation: Practice and Experience*, 19(14), 25 September 2007, pp. 1975-1990.

[24] Viktor Yarmolenko and Rizos Sakellariou. An Evaluation of Heuristics for SLA Based Parallel Job Scheduling. In *Proceedings of the 3rd High Performance Grid Computing Workshop (in conjunction with IPDPS 2006)*, Rhodes, Greece, April 2006, IEEE Computer Society Press.

[25] Jill Dixon and Melany Blackwell. Service Level Agreements – A framework for assuring and improving the quality of support services to faculties. In *Proceedings of the 2002 Annual International Conference of the Higher Education Research and Development Society of Australasia (HERDSA)*, 2002.

[26] Matthew K. O. Lee. IT Outsourcing contracts: Practical Issues for Management. *Industrial Management and Data Systems*, 1996.

[27] M. D'Arienzo, A. Pescapè, S. P. Romano, and G. Ventre. The service level agreement manager: control and management of phone channel bandwidth over Premium IP networks. In *Proceedings of the 15th International Conference on Computer Communication (ICCC02)*, pages 421–432, Washington DC, USA, 2002. International Council for Computer Communication.

[28] Dinesh Verma. *Supporting Service Level Agreements on IP Networks*, Macmillan Technical Publishing, 1999.

[29] Dimitrios Kagklis, Nicolas Liampotis, and Christos Tsakiris. Architecture for the Creation of Service Level Agreements and Activation of IP Added Value Services. In *Proceedings of the 7th International Conference on Telecommunications (ConTEL2003)*, pp. 689-692, 2003.

[30] A. Keller, G. Kar, H. Ludwig, A. Dan, and J. L. Hellerstein. Managing dynamic services: a contract based approach to a conceptual architecture. *Network Operations and Management Symposium (NOMS2002)*, pp. 513-528, 2002.

[31] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. *Agreement-based Grid Service Management (OGSI-Agreement)*. Global Grid Forum, GRAAP-WG Author Contribution, 12 Jun, 2003.

[32] Asit Dan, Catalin Dumitresku, and Matei Ripeanu. Connecting Client Objectives with Resource Capabilities: An Essential Component for Grid Service Management Infrastructures. In *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC04)*, 2004.

[33] D.G.A. Mobach, B.J. Overeinder, and F.M.T. Brazier. A resource negotiation infrastructure for self-managing applications. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing (ICAC 2005)*, Seatle, WA, 2005.

[34] Vijay K. Naik, Swaminathan Sivasubramanian, and Sriram Krishnan. Adaptive Resource Sharing in a Web Services Environment. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Lecture Notes in Computer Science, vol. 3231, pp. 311-330, 2004.

[35] James Padgett, Mohammed Haji, and Karim Djemame. SLA Management in a Service Oriented Architecture. In *Proceedings of the International Conference on Computational Science and its Applications (ICCSA2005)*, Lecture Notes in Computer Science, volume 3483, pp. 1282-1291, 2005.

[36] Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1), March 2003, pp. 57-81.

[37] Heiko Ludwig, Alexander Keller, Asit Dan, Richard King, and Richard Franck. A Service Level Agreement Language for Dynamic Electronic Services. *Electronic Commerce Research*, 3(1-2), January 2003, pp. 43-59.

[38] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck. *Web Service Level Agreement (WSLA) language specification*. IBM Corporation, 2003.

[39] James Skene, D. Davide Lamanna, and Wolfgang Emmerich. Precise Service Level Agreements. *Proceedings of the 26th International Conference on Software Engineering (ICSE2004)*, pp. 179-188.

[40] Lars-Olof Burchard, Matthias Hovestadt, Odej Kao, Axel Keller, and Barry Linnert. The Virtual Resource Manager: An Architecture for SLA-aware Resource Management. In *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid (CCGrid2004)*, pp. 126-133.

[41] Karl Czajkowski, Ian Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *Proceedings of the 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Edinburgh, Scotland, UK, July 24, 2002, Lecture Notes in Computer Science, vol. 2537, pp. 153-183.

[42] Catalin L. Dumitrescu and Ian Foster. GRUBER: A Grid Resource Usage SLA Broker. In *Proceedings of Euro-Par 2005*, Lecture Notes in Computer Science, vol. 3648, pp. 465-474.

[43] Catalin Dumitrescu, Ioan Raicu, Ian Foster. DI-GRUBER: A Distributed Approach to Grid Resource Brokering In *Proceedings of the 2005 ACM/IEEE Supercomputing (SC 2005)*, November 12-18, Seattle, Washington, USA.

[44] Viktor Yarmolenko, Rizos Sakellariou, Djamila Ouelhadj and Jonathan M. Garibaldi. SLA Based Job Scheduling: A Case Study on Policies for Negotiation with Resources. In *Proceedings of the UK e-Science All Hands Meeting 2005 (AHM'2005)*, Nottingham, UK, 19-22 September 2005, CD-ROM Proceedings (Editors: Simon J. Cox, David W. Walker).

[45] Avraham Leff, James T. Rayfield, and Daniel M. Dias. Service-level agreements and commercial grids. *IEEE Internet Computing*, 7(4), July-August 2003, pp. 44-50.

[46] Michael Parkin, Dean Kuo, and John Brooke. A Framework & Negotiation Protocol for Service Contracts. In *Proceedings of the 2006 International Conference on Services Computing (SCC06)*, Chicago, USA, 18-22 September 2006, pp. 253-256.

[47] Michael Parkin, Dean Kuo, John M. Brooke, and Angus MacCulloch. Challenges in EU Grid Contracts. In *Proceedings of eChallenges e-2006 Conference*, Barcelona, Spain, 25-27 October 2006, pp. 67-75.

[48] Sakyibea Darko-Ampem and Maria Katsoufi. Towards A Secure Negotiation Protocol for Virtual Organisations. Master of Science Thesis, Department of Computer and Systems Sciences, Stockholm's University / Royal Institute of Technology, May 2006.

[49] TRUSTCOM EU project.

[50] Peer Hasselmeyer, Henning Mersch, Bastian Koller, H.-N. Quyen, Lutz Schubert, Philipp Wieder. Implementing an SLA Negotiation Framework. *Proceedings of e-Challenges 2007*.

[51] Reid G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, Vol. C-29(12), December 1980, pp. 1104-1113.

[52] Pu Huang and Katia Sycara. A Computational Model For Online Agent Negotiation. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS35'02)*, IEEE Computer Society Press, 2002.

[53] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, M.J. Wooldridge and C. Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2), March 2001, pp. 199-215.

[54] Alexander Barmouta and Rajkumar Buyya. GridBank: a Grid Accounting Services Architecture (GASA) for distributed systems sharing and integration. In *Proceedings of the 2003 International Parallel and Distributed Processing Symposium (IPDPS'03)*, 22-26 April 2003, IEEE Computer Society Press.

[55] Steven Newhouse, Jon MacLaren, and Katarzyna Keahey. Trading Grid services within the UK e-science Grid. In *Grid resource management: state of the art and future trends* (Editors: Jarek Nabrzyski, Jennifer M. Schopf, Jan Weglarz), Kluwer Academic Publishers, pp. 479-490, 2004.

[56] D. Grosu and A. Das. Auctioning resources in Grids: model and protocols. *Concurrency and Computation: Practice and Experience*, vol. 18(15), 2006, pp. 1909-1927.

[57] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Björn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based Grid markets. *Multiagent and Grid Systems*, vol. 1(4), 2005, IOS Press, pp. 297-307.

[58] Liviu Joita, Omer F. Rana, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, and Oscar Ardaiz. Application Deployment on Catallactic Grid Middleware. *IEEE Distributed Systems Online*, 7(12):0612–oz001, 2006.

[59] Zhen Liu, Mark S. Squillante, and Joel L. Wolf. On Maximizing Service Level Agreement Profits. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pp. 213-223, 2001.

[60] Open Grid Forum. http://www.gridforum.org

[61] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. *Web Services Agreement Specification (WS-Agreement)*, version 2005/09. Proposed Recommendation to the Global Grid Forum, 20 September 2005.

[62] Henan Zhao and Rizos Sakellariou. Advance Reservation Policies for Workflows. In *Proceedings of the 12th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, June 2006, Saint-Malo, France. Lecture Notes in Computer Science, volume 4376, pages 47-67.

[63] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1), pp. 41-50, January 2003.

[64] R. Sterritt and M. G. Hinchey. Why Computer-Based Systems Should Be Autonomic. In *Proceedings of the 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2005)*, pp. 406-414, Greenbelt, MD, USA, 3-8 April 2005.

[65] Martyn Fletcher, Jim Austin, and Tom Jackson. Distributed Aero-Engine Condition Monitoring and Diagnosis on the Grid: DAME. In *Proceedings of the 17th International Congress on Condition Monitoring and Diagnostic Engineering Management (COMADEM2004)*, Cambridge, UK, 23-24 Aug, 2004.