Towards Service Level Agreement Based Scheduling on the Grid [*]

Jon MacLaren, Rizos Sakellariou, Krish T. Krishnakumar
University of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

Jon Garibaldi and **Djamila Ouelhadj**
University of Nottingham
Jubilee Campus
Nottingham NG8 1BB
United Kingdom

Abstract

The orchestration of complex workflows on the Grid is emerging as a key goal for the Grid community. It is necessary for these workflows to be executed reliably, respecting any dependences; it is also desirable for the user to know (albeit approximately) when the workflow will complete. The authors argue that meeting these goals will necessitate a major shift in the underlying scheduling technology, which is ultimately used to execute any computational tasks contained in these workflows.

This position paper describes a recently funded project that aims to establish a fundamental new infrastructure for efficient job scheduling on the Grid, based on a notion of Service Level Agreements. These are negotiated between the client (user, superscheduler, or broker) and the scheduler, contain information on acceptable job start and end times, and may be re-negotiated during runtime.

Introduction

Within the Grid community at present, there is a keen focus on the management and scheduling of workflows, i.e. complex jobs, consisting multiple computational tasks, connected either in a Directed Acyclic Graph (DAG), or in a more general graph, incorporating conditionals and branches. In order for a workflow enactment engine, such as the GriPhyN Project's Pegasus (PEGASUS 2004), or the UNICORE Grid middleware (UNICORE 2004), to successfully orchestrate these workflows, it must be possible to schedule multiple computational tasks onto (possibly) distributed resources, while still respecting any dependences in the workflow. Current methods employed to schedule work on compute resources within the Grid are unsuitable for this purpose.

Although there has been a considerable amount of work related to scheduling DAGs onto heterogeneous systems (Sakellariou & Zhao 2004; Topcuoglu, Hariri, & Wu 2002) this is not directly applicable in the Grid context since it assumes a static environment where execution costs are well-known in advance; only limited attempts have been made to consider situations where run-time changes in the estimated costs may require some reconsideration of scheduling decisions (Zhao & Sakellariou 2004). As a result, at the moment, the enactment of a workflow has to rely on traditional, queue-based scheduling systems. Such scheduling systems (often called "batch" schedulers) are queue-based, and provide only one level of service, namely 'run this when it gets to the head of the queue', which approximates to 'whenever'. This uncertainty means that any workflow enactment engine must wait for components of the workflow to complete before *beginning* to schedule dependent components. This approach fails to hide the latencies resulting from the length of the job queues, which then control and determine the execution time of the workflow.

New patterns of usage arising from Grid computing (and other areas) have resulted in the introduction of advance reservation to these schedulers, where jobs can be made to run at a precise time. However, this is also an extreme level of service, and is excessive for many workflows, where often it would be sufficient to know the latest finish time, or perhaps the soonest start time and latest end time. Advance reservation (in its current form) is also unsuitable for any scenario involving the simultaneous scheduling of multiple computation tasks, either as a sequence, or tasks that must be co-scheduled. In such cases, a client (superscheduler or broker) must be able to simultaneously negotiate with a number of resources, only committing to suitable arrangements. Current advance reservation APIs only allow the request of a reservation, with a yes/no answer, where 'yes' denotes a booking. This is inadequate, as good guessing is required to book all the resources for suitable times, the difficulty increasing exponentially with the number of resources.

Also, in its current form, advance reservation has several disadvantages for the resource owner. When an advance reservation is made, the scheduler must place jobs around this fixed job. Typically, this is done using backfilling (Lifka 1995), which increases utilisation by searching the work queues for small jobs, which can 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

[*]This work is funded by the EPSRC Fundamental Computer Science for e-Science initiative (Grant GR/S67654/01), whose support we are pleased to acknowledge.

total memory footprint of 3Gb; the checkpointing of capability 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. As these new patterns of usage increase, utilisation will fall further. 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.

These two current levels of service, 'run this job whenever' and 'run this job at this precise time' are two extremes on a scale. A recently funded project aims to establish a fundamental new infrastructure for efficient job scheduling on the Grid, based on a notion of Service Level Agreements; these are negotiated between the client (user, superscheduler, or broker) and the scheduler, contain information on acceptable job start and end times, and may be re-negotiated during runtime. The approach followed in the project, which allows a free exploration of the space between the aforementioned extremes, is briefly described below.

Approach

Service Level Agreements (SLAs) (Keller *et al.* 2002) are emerging as the standard concept by which work on the Grid can be arranged and co-ordinated. An SLA is a bilateral agreement, typically between a service provider and a service consumer. These form a natural choice for representing the agreed constraints for individual jobs. While there are technologies for composing SLAs in XML-based representations, e.g. WSLA (Ludwig *et al.* 2002), these embed domain-specific terms; no terms for resource reservation have yet been proposed within the Grid community. In any case, it is certain that SLAs can be designed to include acceptable start and end time bounds and a simple description of resource requirements. SLAs expressing conventional requirements of "at time HH:MM" or "whenever" could still be used where necessary, although these—especially the latter—may not be accepted by the scheduler. The GGF GRAAP Working Group (GRAAP 2004) is interested in SLA terms for resource reservation, but has not yet put forward any designs for what these SLAs should look like. It is intended that the project will feed back a design of these SLA terms to the community, contributing to the standardisation process. The project team expects to be able to use existing negotiation schemes such as Contract Net (Smith 1980), while also contributing to and influencing the development of emerging technologies, such as the negotiation protocol WS-Agreement (Czajkowski *et al.* 2003a), which is being defined by the GGF GRAAP Working Group.

The Contract Net protocol is a high-level protocol for achieving efficient cooperation in agent-based systems (Ferber 1999; Wooldridge 2002) based on a market-like protocol. In this approach, a manager agent advertises a task to execute by a task announcement to other agents in the net. In response, contracting agents evaluate the task with respect to their abilities and engagements and submit bids. The manager agent evaluates the submitted bids and selects the most appropriate bidder to execute the task, which leads to awarding a contract to the contractor with the most appropriate bid.

Advantages of the Contract Net protocol include the following: dynamic task allocation via self-bidding which leads to better agreements; natural load balancing as busy agents need not bid; flexibility as agents can be introduced and removed dynamically; and reliability since it provides a robust mechanism for distributed scheduling and failure recovery.

Scheduling algorithms that offer a fundamentally more advanced degree of flexibility will be required in order to take full advantage of both the new timing information, and of the SLA negotiation and re-negotiation processes. These algorithms will have to optimize with standard criteria, such as resource utilisation, but they must also be able to evaluate a proposed SLA with respect to its current set of accepted SLAs, and be able to indicate whether or not the SLA was acceptable, returning a proposed SLA with additional information in the affirmative case, or with suggested altered parameters in the negative case; this forms the scheduler's ability to negotiate.

Scheduling models based on fuzzy methods have recently attracted interest among the scheduling research community (Slowinski & Hapke 2000). Often, scheduling research employs a single evaluation function based on how well a schedule meets a combination of various constraints, but this approach overlooks the fact that a user may wish to evaluate a schedule by multiple criteria (e.g., the user may prefer an SLA offering high-cost / quick and reliable turnaround over an alternative offering low-cost / uncertain turnaround). The proposed scheduler will have the capability of efficiently matching flexible user requirements expressed within SLAs with available resources, initially based on the assumption of perfect information and then extended to incorporate fuzzy, multi-criteria approaches.

The project will also consider the renegotiation of resources by running jobs. Re-negotiation has been identified as a long-term goal of the RealityGrid UK e-Science Pilot Project (Czajkowski *et al.* 2003b, Sec. 4.2); here, simulations may be collaboratively visualized and steered as the simulation runs. Renegotiation is required for multiple reasons: to extend the running time of an increasingly interesting simulation; to increase the computational resources dedicated to either the simulation, thereby accelerating the experiment, or to the visualization, in order to improve the resolution of the rendering; resources might also be given back when the improved speed or picture quality was no longer required. Also, more generally, resources may fail unpredictably, high-priority jobs may be submitted, etc. In a busy Grid environment, SLAs would be constantly being added, altered or withdrawn, and hence scheduling would need to be a continual, dynamic and uncertain process. The introduction of re-negotiation, permitted during job execution (as well as before it commences) makes the schedule more dynamic, requiring more frequent rebuilding of the schedule.

Shift in Problem Domain

In attempting to guarantee start and finish bounds for all jobs, the scheduling problem becomes significantly changed. In the traditional batch processing model, jobs run until they fall over, finish, or run over time and are killed by the scheduler. As there is little advance planning required, when a job completes far ahead of time, the schedule can be revised; there is no impact on other jobs, as there is no expected Quality of Service. However, if the scheduler has made assurances regarding each job in the system, the early termination of a job could result in either resources being left idle, or SLAs being broken. To avoid this, the scheduler may try to ensure that there are always a number of "flexible" SLAs in its schedule, i.e. those with easily-met constraints. In addition, the scheduler must also allow for uncertainties in the data supplied by the user, e.g. expected run time.

Given the fact that there is always a significant percentage of jobs which fall over immediately (due to problems stemming from the job submission script, such as, syntax errors, incorrectly specified files, etc), the scheduler would likely employ overbooking to some extent. There are many similarities between this problem and scenarios such as airline seat and hotel room booking. However, there are two features to this problem that make it more complex than either of these scenarios, placing it on the cutting edge of scheduling research: first, the permitted renegotiation of resources, which makes the problem very dynamic; and secondly, the time constraints in which the schedule must be solved.

A scheduler will also want to retain some flexibility so that it can accept short-notice jobs, for which a high price would be charged. It is also possible that a scheduler would decide to break a few smaller or cheaper SLAs in order to get such work. Regardless of the optimisation objective in such cases, something that the scheduler would have to cope with, a model of client behaviour might also be needed to retain flexibility.

Conclusion

To conclude, the enormous potential of the Grid cannot be fully realised until fundamental development of powerful new scheduling algorithms has taken place. This project will investigate and develop novel scheduling approaches which will address the critical issues of flexibility and renegotiation of user requests and will seek to address the problem of handling uncertainty and imprecision in both compute resources and user requirements. The method will have to negotiate a Service Level Agreement with the submitter of each job; the SLAs will include the possibly imprecise acceptable job start and end times. This paradigm shift will enable users and super-schedulers alike to successfully schedule and orchestrate complex multi-part Grid jobs across multiple resources.

References

Czajkowski, K.; Dan, A.; Rofrano, J.; Tuecke, S.; and Xu, M. 2003a. Agreement-based Service Management (WS-Agreement). *Draft Global Grid Forum Recommendation Document*.

Czajkowski, K.; Pickles, S.; Pruyne, J.; and Sander, V. 2003b. Usage Scenarios for a Grid Resource Allocation Agreement Protocol. *Draft Global Grid Forum Informational Document*.

Ferber, J. 1999. *Multi-agent systems: Introduction to Distributed Artificial Intelligence*. London: Addison-Wesley.

GRAAP. 2004. GRAAP-WG, Grid Resource Allocation Agreement Protocol Working Group in the Global Grid Forum. Website: `https://forge.gridforum.org/projects/graap-wg/`.

Keller, A.; Kar, G.; Ludwig, H.; Dan, A.; and Hellerstein, J. L. 2002. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. *Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002)* 513–528.

Lifka, D. A. 1995. The ANL/IBM SP Scheduling System. *Proceedings of the IPPS '95 Workshop "Job Scheduling Strategies for Parallel Processing"* 949:295–303.

Ludwig, H.; Keller, A.; Dan, A.; and King, R. 2002. A Service Level Agreement Language for Dynamic Electronic Services. *Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002)* 25–32.

PEGASUS. 2004. The Pegasus Project: Planning for Execution in Grids. Website: `http://www.isi.edu/~deelman/pegasus.htm`.

Sakellariou, R., and Zhao, H. 2004. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. *Proceedings of the 13th International Heterogeneous Computing Workshop*.

Slowinski, R., and Hapke, M. 2000. *Scheduling under Fuzziness*. Physica Verlag.

Smith, R. G. 1980. The Contract Net Protocol – High-level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers* 29(12):1104–1113.

Topcuoglu, H.; Hariri, S.; and Wu, M. 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems* 13(3):260–274.

UNICORE. 2004. The UNICORE Grid Middleware. Available from the UNICORE Forum website: `http://www.unicore.org/`.

Wooldridge, M. 2002. *An Introduction to Multi-agent Systems*. Winchester, England: John Wiley and Sons Ltd.

Zhao, H., and Sakellariou, R. 2004. A Low-Cost Rescheduling Policy for Dependent Tasks on Grid Computing Systems. *Proceedings of the 2nd Across Grids Conference*.