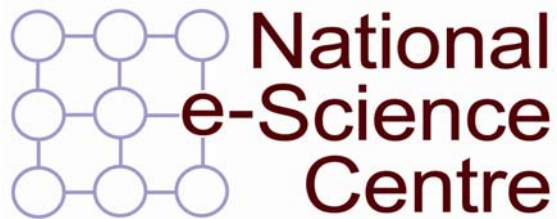


Open Issues in Grid Scheduling

Report of the workshop held at the e-Science Institute, October 21-22nd 2003

Alain Andrieux, Globus Alliance
Dave Berry, National e-Science Centre
Jon Garibaldi, University of Nottingham
Stephen Jarvis, University of Warwick
Jon MacLaren, University of Manchester
Djamila Ouelhadj, University of Nottingham
Dave Snelling, Fujitsu Labs Europe



the Inter-disciplinary
Scheduling Network

This report is available online at

http://www.nesc.ac.uk/technical_papers/UKeS-2004-03.pdf

Electronic copies of presentations made at this workshop are available at

<http://www.nesc.ac.uk/action/esi/contribution.cfm?Title=309>

Contents

1.	Introduction.....	3
1.1.	Overview.....	3
1.2.	Motivation for the Workshop.....	3
1.3.	Structure.....	4
2.	Breakout Session: Grid Scheduling and Traditional Scheduling.....	5
2.1.	Introduction.....	5
2.2.	Terminology.....	5
2.3.	The state of the art: A review of traditional scheduling methodologies.....	6
2.4.	References.....	9
3.	Breakout Session: A Broker/Scheduler Architecture for Grid Services.....	12
3.1.	Introduction.....	12
3.2.	Using WS-Agreement.....	12
3.3.	Broker Architecture Decomposition and Interaction Model.....	15
3.4.	Hierarchy of brokers.....	17
3.5.	Conclusion.....	18
3.6.	References.....	18
4.	Performance.....	20
4.1.	General Issues.....	20
4.2.	Performance metrics.....	22
4.3.	Algorithms.....	23
4.4.	References.....	23
5.	Workshop Presentations.....	25
5.1.	Session One: Current Grid Work.....	25
5.2.	Session Two: New Ways to Schedule.....	26
5.3.	Session Three: Insights From Traditional Scheduling.....	27
5.4.	Overview: A Broker/Scheduler Architecture for Grid Services.....	28
5.5.	Overview: Service Level Agreement Based Scheduling.....	29

1. Introduction

1.1. Overview

This report summarises the workshop on *Open Issues in Grid Scheduling* that was held at the UK e-Science Institute on October 21-22nd 2003. It was co-hosted by the National e-Science Centre (www.nesc.ac.uk) and the EPSRC Inter-disciplinary Scheduling Network (<http://www.asap.cs.nott.ac.uk/iol/is-network/index.shtml>). The aim of the workshop was to identify open issues in the field of scheduling activities on computation Grids.

Forty people attended from around the world, including researchers who approach these issues from different backgrounds. Topics discussed included planning, failure handling and recovery, optimisation and re-optimisation, data movement and co-location, scheduling policies and economic models of resource provision. This report describes the problems identified and suggests possible approaches to addressing them.

1.2. Motivation for the Workshop

At some point in the past, probably in the 1960s, "scheduling" became two separate research areas, with people looking at the scheduling of processes, jobs, etc. onto CPUs and computers separating off from people concerned with optimising timetables, organising factory workflows, etc. Scheduling on the Grid presents an excellent opportunity to re-unite these long-separated siblings.

Many of those in the early Grid Community were from a supercomputing background; they were interested in computational Grids, focussing on problems like "How do I make running a job on all these different resources the same task? And with the same authentication mechanism?" Precisely the sort of problems that led Argonne and ISI to develop the Globus Toolkit.

But now, as different communities with different requirements and objectives arrive in the Grid Community, the focus is changing. People are thinking about orchestrating workflows in a service-based environment. People need to be able to dispatch work, but with assurances about when the work will be completed, performance and cost, etc., all negotiated as part of some Service Level Agreement. So the designer of a Grid-based scheduler component would need to think "How can we best schedule this work onto these resources, given that each job has an agreed set of constraints, so that we meet as many constraints as possible, while still trying to maximise income?" Currently, the Grid Community is unsure as to how to solve this problem. Why? Because it's a typical "traditional scheduling" problem; and the people who know how to solve such problems are not currently engaged in the Grid Community.

As the Grid evolves, Grid Scheduling problems are starting to look more and more like "traditional scheduling" problems. This workshop provided an arena for the Grid Scheduling community and the current scheduling research community to get together, exchange ideas, and foster potential collaborative research.

1.3. Structure

The first day of the workshop consisted of presentations by researchers active in various aspects of scheduling. The presentations were grouped into three sessions: *Current Grid Work*, *New Ways to Schedule*, and *Insights from Traditional Scheduling*.

The second day was devoted to two parallel breakout discussions and their reports to the whole group. One breakout focussed on Dave Snelling's *A Broker/Scheduler Architecture for Grid Services*. The other looked at *Grid Scheduling and Traditional Scheduling*, and was led by Jon Garibaldi.

This report is structured as follows. The next two sections present the outcomes of the two breakout sessions. These are followed by a discussion of performance issues. Finally the report lists the presentations given at the workshop, in some cases including their abstracts.

2. Breakout Session: Grid Scheduling and Traditional Scheduling

Chair of the session: Jon Garibaldi, University of Nottingham

2.1. Introduction

The breakout session on *Grid Scheduling and Traditional Scheduling* identified several critical issues in the field of scheduling activities on Computation Grids. In this report we raise these scheduling issues and give some approaches already adopted in traditional scheduling which could be investigated in Grid Scheduling.

2.2. Terminology

The scheduling terminology used by the Grid community seems quite different from the one used by the traditional scheduling community. We have listed some basic definitions, which may clarify or expand the Grid scheduling terminology:

What is a **task**: a task is the atomic operation to be performed on the resource.

What are the **properties** of a task: deadline, priority, etc.

What is a **job**: a job is single set of multiple atomic operations that will be carried out on a set of resources. Jobs can have recursive structure, meaning that jobs are composed of sub-jobs and/or tasks, and sub-jobs can themselves be decomposed further and tasks are the leaves in such a tree.

What is a **resource**: a resource is something that is required to carry out an operation. Examples: disk space, CPU time, etc.

What is **scheduling**: scheduling is the mapping of tasks to resources.

What is a **workflow**: it is the ordering of a set of jobs for a specific user.

2.2.1. A Traditional Scheduling View of Grid Scheduling

In traditional scheduling, scheduling is defined as the allocation of operations to resources over time, taking into account some performance measure criteria subject to the satisfaction of constraints [Pinedo, 1995]. So, is Grid scheduling a new problem which differs from the traditional scheduling? There is no clear evidence that it differs in any fundamental way from the traditional scheduling problems. The fundamental difference may be the dynamic nature of resources and constraints in the Grid environment, but this point is not clear at present.

Grid scheduling: grid scheduling is the mapping of individual tasks to computer resources, while respecting service level agreements (SLAs), etc.

Constraints: scheduling constraints can be hard or soft. Hard constraints are rigidly enforced. Soft constraints are those that are desirable but not absolutely essential. Soft constraints are usually combined into an objective function. So, what are the hard and soft constraints in Grid scheduling?

Optimisation Criteria: a variety of optimisation criteria are of interest for Grid scheduling: minimisation of the maximum lateness, minimisation of the cost to the user, maximisation of the profit, maximisation of personal or general utility, maximisation of resource utilisation, fairness, minimisation of variance, maximisation of robustness and predictability, minimisation of broken SLAs, etc.

Scheduling data: a variety of data are necessary for a scheduler to describe the jobs and the resources: job length, resource requirements estimates, time profiles, uncertain estimate, etc.

Methodologies: the meeting agreed that there is no clear choice of method which will be the best for Grid scheduling.

2.3. The state of the art: A review of traditional scheduling methodologies

Grid scheduling could benefit from several traditional scheduling methodologies. These methodologies have achieved successful results in a wide range of scheduling applications. Therefore, it is worth to start to investigate their performance in Grid scheduling. These methodologies are described below.

2.3.1. Heuristics:

A Heuristic is a technique that seeks good solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is [Reeves, 1995]. Dispatching rules are examples of heuristics; they are used to select the next job to process on the resource whenever the resource becomes free. Dispatch rules include EDD (Earliest Due Date) and FCFS (First Come First Served).

2.3.1.1. Meta-heuristics: tabu search, simulated annealing and evolutionary algorithms

Meta-heuristics are high-level heuristics that guide local search heuristics to escape from local optima. Meta-heuristics such as tabu search, simulated annealing and genetic algorithms improve the local search algorithms to escape local optima by either accepting worse solutions, or by generating good starting solutions for the local search in a more intelligent way than just providing random initial solutions [Reeves, 1995, 1997; Reeves and Rowe, 2002; Voss et al., 1999; Pham and Karaboga, 2000].

2.3.1.2. Knowledge-based systems

Knowledge-based systems focus on capturing the expertise or the experience of the scheduling expert and an inference mechanism is used to derive conclusions or recommendations regarding the scheduling problem [Szelke and Kerr, 1994; Zweben and Fox, 1994; Weirs, 1997].

2.3.1.3. Case-based reasoning

Case-Based Reasoning (CBR) is an artificial intelligence methodology in which a new problem is solved by reusing knowledge and experience gained in solving previous

problems. A case contains a description of the problem, and its solution. Cases are stored in a case base. The CBR process is divided into four phases: retrieval of the case most similar to the new problem, reuse and revision of its solution, and inclusion of the new case in the case base [Kolodner, 1993; Aamodt and Plaza, 1994; Leake, 1996].

2.3.1.4. Dynamic scheduling

Dynamic scheduling is the problem of scheduling in dynamic environments. Grid scheduling systems operate in dynamic environments subject to various unforeseen and unplanned events that can happen at short notice. Such events include the breakdown of computers, arrival of new jobs, processing times are subject to stochastic variations, etc. It turns out that the performance of a schedule is very sensitive to these disturbances, and it is difficult to execute a predictive schedule generated in advance. These real-time events not only interrupt system operation but also upset the predictive schedule that was previously established. Consequently the resulting schedule may neither be feasible nor nearly optimal anymore. Dynamic scheduling is arguably of practical importance in Grid Scheduling to generate robust schedules. For extensive surveys on dynamic scheduling refer to [Cowling and Johanson, 2002; Cowling et al., 2003a; Vieira et al., 2003].

2.3.1.5. Fuzzy methodologies

Fuzzy systems consist of a variety of concepts and techniques for representing and inferring knowledge that is imprecise, uncertain, or unreliable. Scheduling models based on fuzzy methods have recently attracted interest among the scheduling research community [Slowinski and Hapke, 2000]. A fuzzy set is a very general concept that extends the notion of a standard set defined by a binary membership to accommodate gradual transitions through various degrees. Since the original introduction of fuzzy sets by Lotfi Zadeh in 1965 [Zadeh, 1965], the notion has been extended to a complete framework of fuzzy methodology incorporating aspects such as fuzzy numbers, fuzzy arithmetic and fuzzy relations [Ruspini et al., 1998], and fuzzy reasoning [Zadeh, 1975].

Previous work has investigated the representation of uncertainty in processing time and due time by fuzzy numbers, the representation of flexible constraints by fuzzy measures, fuzzy job precedence relations or machine breakdowns, but these have been in isolation. Even once an SLA has been agreed, there are many ways in which it might need re-negotiation: (compute and other) resources may fail unpredictably, sub-jobs may fail due to user error, more important (high-priority) jobs may be submitted, user-requirements might change, 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. Some preliminary work has been carried out to examine whether fuzzy methods can be used in the evaluation of Grid *performance contract violations* [Vraalsen et al., 2001], but this has been very simplistic (based on 2 fuzzy rules with 2 variables).

2.3.1.6. Agents and multi-agent systems

Recently, multi-agent systems are one of the most promising approaches to building complex, robust, and cost-effective next-generation manufacturing scheduling systems

because of their autonomous, distributed and dynamic nature, and their robustness against failures.

An agent is a computer system that is situated in some environment, and that is capable of flexible and autonomous action in this environment in order to meet its design objectives. By flexible we mean that the system must be responsive, proactive, and social [Wooldridge and Jennings, 1995].

A Multi-Agent System is a system composed of a population of autonomous agents, which interact with each other to reach common objectives, while simultaneously each agent pursues individual objectives [Oliveira et al., 1998].

The motivations for the increasing interest in multi-agent based scheduling research can be explained in the following points [Parunak, 1996; Sousa and Ramos, 1999; Shen and Norrie, 1999, 2001; Cowling et al., 2000].

- Real-life scheduling problems are usually physically or functionally distributed (air traffic control, manufacturing systems, health care, etc.).
- Complex scheduling systems are beyond direct control. They operate through the co-operation of many interacting subsystems, which may have their independent interest, and modes of operation.
- The complexity of practical scheduling problems dictates a local point of view. When the problems are too extensive to be analysed as a whole, solutions based on local approaches are more efficient.
- Centralised scheduling structures are difficult to maintain and reconfigure, inefficient to satisfy real-world needs, and costly in the presence of failures. Also, the amount of knowledge to manage is very large.
- There is a need for integration of multiple legacy systems and expertise.
- Real-world scheduling problems are heterogeneous. Heterogeneous environments may use different data and models, and operate in different modes.
- Agents provide robustness and reliability against failures. Distributed systems allow fast detection and recovery from failures, and the failure of one or several agents does not necessarily make the overall system useless.
- Because multi-agent systems are open and dynamic structures, the system can be adapted to an increased problem size by adding new agents, and this does not affect the functionality of the other agents.
- Agents can operate asynchronously and in parallel, which can result in increased overall speed.
- Individual agents can be developed separately and it may be possible to reuse agents in different application scenarios. Moreover, the overall system can be tested and maintained and reconfigured more easily.
- Agents may be much more cost-effective than a centralised system, since it could be composed of simple subsystems of low unit cost.

Multi-agents have been successfully used to resolve a wide range of complex distributed scheduling problems [Cowling et al., 2001, 2003a, 2003b; Ouelhadj et al., 1998, 1999, 2000, 2003a, 2003b]. We believe that Multi-agent systems provide the foundation for the creation of Grid scheduling systems that possess capabilities of autonomy, heterogeneity,

reliability, maintainability, flexibility, and robustness. Some researchers have already started to investigate the use of the agent technology in grid computing, in particular for resource management in grid environments [Cao et al., 2002a, 2002b].

For more details on the scheduling methodologies refer to the ASAP brochure on http://www.asap.cs.nott.ac.uk/ASAP_Brochure2002.pdf

2.4. References

Aamodt, A., Plaza, P. (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches*. The European Journal on Artificial Intelligence, 7(1): 39-59.

Cao, J., Spooner, D. and Nudd G. (2002a) *Agent-based resource management for grid computing*. Available at <http://www.dcs.warwick.ac.uk/~hpsg/html/downloads/public/docs/CaoJ.ARMGC.pdf>.

Cao, J., Jarvis, S. A., Saini S. et al. (2002b) *ARMS: an agent-based resource management system for grid computing*. Scientific Programming, 10(2), 135-148.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2000) *Multi-agent systems for dynamic scheduling*. In: Proceedings of the Nineteenth Workshop of Planning and Scheduling of the UK, PLANSIG 2000, pp. 45-54, Ed. Garagnani, Max, The Open University, UK.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2001) *A Multi-agent architecture for dynamic scheduling of steel hot rolling*. In: Proceedings of the Third International ICSC World Manufacturing Congress, Rochester, NY, USA, pp. 104-111.

Cowling, P. I. and Johansson, M. (2002) *Using real time information for effective dynamic scheduling*, European Journal of Operational Research, 139 (2), 230-244.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2003a) *A Multi-agent architecture for dynamic scheduling of steel hot rolling*. Journal of Intelligent Manufacturing, 14, 457-470.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2003b) *Dynamic scheduling of steel casting and milling using multi-agents*. To appear in the Journal of Production Planning and Control, Special Issue on the Application of Multi Agent Systems to Production Planning and Control.

Leake D.B. (1996) *CBR in Context: The Present and Future*. In Leake D. (ed.): Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press/MIT Press,

Kolodner J. (1993) *Case-Based Reasoning*. Morgan Kaufmann Publishers.

Oliveira, E., Fischer, K. and Stepankova, O. (1998) *Multi-agent systems: which research for which applications*. Robotics and Autonomous Systems, 7 (1-2), 91-106.

Ouelhadj, D., Hanachi, C. and Bouzouia, B. (1998) *Multi-agent system for dynamic scheduling and control in manufacturing cells*. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1256-1262, Belgium.

Ouelhadj, D., Hanachi, C. and Bouzouia, B. (1999) *A Multi-contract net protocol for dynamic scheduling in flexible manufacturing systems*. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1114-1120, USA.

- Ouelhadj, D., Hanachi, C. and Bouzouia, B. (2000) *Multi-agent architecture for distributed monitoring in flexible manufacturing systems (FMS)*. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1120-1126, San Francisco, USA.
- Ouelhadj, D., Cowling, P. I., and Petrovic, S. (2003a) *Contract net protocol for cooperative optimisation and dynamic scheduling of steel production*. In: Intelligent Systems Design and Applications, pp. 457-470, Eds. Ajith, Ibrahim, Katrin, Franke and Mario, Koppen, Springer-Verlag.
- Ouelhadj, D., Cowling, P. I. and Petrovic, S. (2003b) *Utility and stability measures for agent-based dynamic scheduling of steel continuous casting*. In: Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 175-180.
- Parunak, H. V. (1996) *Applications of distributed artificial Intelligence in industry*. In: Foundation of Distributed Artificial Intelligence, Chapter 4, Eds. O'Hare, G. M. P. and Jennings, N. R., Wiley Inter-science, New York.
- Pham, D. T. and Karaboga, D. (2000) *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer, London.
- Pinedo, M. (1995) *Scheduling theory, algorithms and systems*. First edition, Prentice Hall.
- Reeves, C. R. (1995) *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, McGraw-Hill International (UK) Limited.
- Reeves, C.R. (1997) *Genetic algorithms for the Operations Researcher*. INFORMS J on Computing, 9, 231-250.
- Reeves, C.R. and Rowe, J.E. (2002) *Genetic Algorithms—Principles and Perspectives*, Kluwer Academic Publishers, Norwell, MA.
- Ruspini, E., Bonissone, P., Pedrycz, W., (Eds.) (1998), *Handbook of Fuzzy Computation*, Institute of Physics Publishing, Bristol and Philadelphia.
- Shen, W. and Norrie, D. H. (1999) *Agent based systems for intelligent manufacturing: a state of the art survey*. International Journal of Knowledge and Information Systems, 1 (2), 129-156.
- Shen, W., Norrie, D. H. and Barthes, J. P. A. (2001) *Multi-agent systems for concurrent intelligent design and manufacturing*. Taylor & Francis, London.
- Slowinski, R., and Hapke, M., (Eds.) (2000), *Scheduling Under Fuzziness*, Physica-Verlag.
- Sousa, P. and Ramos, C. (1999) *A distributed architecture and negotiation protocol for scheduling in manufacturing systems*. Computers in Industry, 38 (2), 103-113.
- Szelke, E. and Kerr R. M. (1994) *Knowledge-based reactive scheduling*. North-Holland.
- Vieira, G. E., Hermann, J. W. and Lin, E. (2003) *Rescheduling manufacturing systems: a framework of strategies, policies and methods*. Journal of Scheduling, 6 (1), 36-92.
- Voss, S., Roucairol, C. and Osman, I. H. (1999) *Meta-Heuristics: Advances and trends in local search paradigms for optimization*. Kluwer Academic Publishers.

Vraalsen, F., Aydt, R.A., Mendes, C.L. and Reed, D.A. (2001), *Performance Contracts: Predicting and Monitoring Grid Application Behavior*, in Proceedings of the 2nd International Workshop on Grid Computing (GRID 2001), Lecture Notes in Computer Science (LNCS) Vol. 2242, pp. 154-165, Springer-Verlag.

Weirs, V. C. S. (1997) *A review of the applicability of OR and AI scheduling techniques in practice*. Omega International Journal of Management Science, 25 (2), 145-153.

Wooldrige, M. and Jennings, N. R. (1995) *Intelligent agents: theory and practice*. Knowledge Engineering Review, 10 (2), 115-152.

Zadeh, L.A. (1965). *Fuzzy Sets*. Information and Control, 8, 338-353.

Zadeh, L.A. (1975). *The Concept of a Linguistic Variable and its Application to Approximate Reasoning; Parts I, II and III*. Information Sciences, 8, 199-249; 8, 301-257; 9, 43-80.

Zweben, M. and Fox, M. S. (1994) *Intelligent scheduling*. Morgan Kaufmann Publishers, INC.

3. Breakout Session: A Broker/Scheduler Architecture for Grid Services

Chair of the session: Dave Snelling, Fujitsu Laboratories of Europe

3.1. Introduction

This breakout session discussed the "Broker/Scheduler Architecture for Grid Services" presented David Snelling from Fujitsu Labs Europe (see 5.4)

The discussion had two parts. Firstly, the WS-Agreement specification [Czajkowski et al., 2003] was discussed in order to see how it could be applied to the architecture.

Then, we discussed the decomposition of the broker architecture into different interacting software actors, such as implementation objects or exposed services (the differentiation in the model being undefined at this point). In particular we tried and checked that the "factoring" of functionalities made sense in the model, i.e. that there is a clean distribution of responsibilities.

3.2. Using WS-Agreement

The WS-Agreement specification, still in infancy, is a promising approach to provisioning services in an interoperable manner. Its negotiation features make it an interesting framework to use for schedulers to advertise resource capabilities and for clients to request computational job requests. There is agreement that it would be interesting and useful to use WS-Agreement at the interface of schedulers and brokers in the architecture. However, one must be sure the requirements of the broker architecture are addressed by the specification, and that some concerns are alleviated.

It is important to point out that WS-Agreement is still very much in the making, and as a specification it needs many more rounds of discussion in order to reach a state that is satisfactory for actual use. The Global Grid Forum (GGF) working group called GRAAP [GRAAP-WG] is in charge of handling such discussions and the evolution of the specification. In addition, it fills the role of a point of contact for asking questions about WS-Agreement and/or requesting features.

3.2.1. Modelling batch job scheduling

WS-Agreement is a domain-independent specification. In order to apply it to the broker/scheduler architecture, one must first wonder how to apply it to job submission and control. The answer is domain-specific modelling at two levels:

3.2.1.1. Service Port Types

Job submission port types (Web Service interfaces written in WSDL) may need to be defined by extending the domain-independent port types exposed in the WS-Agreement specification.

For instance, a JobAgreement port type, while encapsulating the terms of agreement negotiated between the scheduler and the client, would be the service interface to a job

entry in a queue (a JobAgreement service representing a job entry). The JobAgreement port type would in fact provide the client with an interface definition for control of the job via operations (start, suspend, resume, etc) and/or for monitoring of the execution via service data.

3.2.1.2. Domain-specific Terms

However, the major piece of work in a domain-specific extension of WS-Agreement is to define domain-specific agreement terms. Here, what needs to be defined is a term language to describe job requirements and attributes that can be negotiated by clients and schedulers to reach agreements.

There would be great benefits to the community in standardizing such a domain-specific language, therefore the process of defining the term language should be the role of a standard initiative. The JSDL working group of GGF [JSDL-WG] seeks to define a description language for computational jobs. The JSDL-WG is currently looking at the possibility of using the WS-Agreement term meta-language (still in discussion) as a framework for writing the JSDL, and WS-Agreement interface mechanisms as the communication protocol for submission and control (if a job term language is standardized, job submission interfaces may need to be standardized as well). An attempt for a term meta-language exists in a proposal [Andrieux et al., 2003] that was made by Globus and Platform to the JSDL-WG. The proposal features domain-specific terms for job description with examples showing how to write domain-specific terms using domain-independent constraint operators.

3.2.1.3. A comment on the meta-language:

The WS-Agreement specification is to specify a domain-independent term language that enables service providers and consumers to express domain-specific terms with markup such as constraints for negotiation and/or for service semantics. By constraints we not only mean fixed values but also inequalities (lower and/or upper bounds). In fact, while it can be difficult for both parties in a negotiation to reach agreement on fixed values, the flexibility expressed by constraints such as boundaries can considerably ease the negotiation process and enable parties to agree on constraints that satisfy them both. For instance, in the domain of computational jobs, a term of agreement between a scheduler and a client could be the time by which the job must be dispatched. Negotiating a fixed value on that term can be difficult if not impossible, because the scheduler cannot guarantee a fixed time for every possible job. Also, the client may not be interested in a fixed time but more in a temporal boundary. On the other hand, the scheduler may be able to guarantee a maximum time for the job to be dispatched - a boundary that would depend on the policies of the scheduler for scheduling jobs in the community, and what it is willing to negotiate with clients. This flexibility brought by quality of service constraints was suggested by several presentations made during the seminar (see e.g. 5.2.5, 5.5) as a solution to the conflict between the competing interests of the resource users and the resource owners and find materialization in the WS-Agreement term language.

3.2.2. Requirement for client/scheduler symmetry

A resource management requirement is that schedulers whose resources are idle be able to actively request clients for jobs to be requested, so as to optimize the utilization of the resources.

The issue raised here then is: how can a scheduler bid for jobs through a WS-Agreement façade? In WS-Agreement a service provider advertises its capabilities as part of its dynamic interface using dynamic service data. From the discussion, the requirements for bilateralism/symmetry of communication between the two parties seem to go beyond the mere advertisement of acceptable agreements.

As a result of the workshop described in this report, this question was raised on the GRAAP mailing-list. The answer has several parts. Firstly, the asymmetry of WS-Agreement as a protocol was carefully chosen to reduce the operational effort needed to become an initiator/client, i.e. to prevent from having to make the initiator a dedicated Grid service. Let's note that the agreement binds both parties, with certain terms putting the responsibility on one party and other terms on the other party. For instance, only a provider can guarantee a start time (or upper bound thereof) for a submitted job, but only the consumer can specify the amount of memory the job will use for execution. On the other hand, the provider has the possibility to initiate communication with the initiator via asynchronous notifications, provided the initiator has subscribed to them. It is also possible to design the client as a provider as well and then have the scheduler act as an initiator negotiating with this new provider. In this way there is no need to define another client/server protocol than the one already specified, but again, this may be too heavyweight for most cases compared to simple notifications sent to the client. As a conclusion, the requirement is addressed by WS-Agreement, but parts of the specification need to be re-phrased in a next revision so as to make these points more understandable.¹

3.2.3. Miscellaneous questions

In this section we cover other questions that were asked during the discussion.

3.2.3.1. Rejection of request

Question: When an invalid request/offer for an agreement is rejected, is the client explained what part of the agreement offer was rejected?

Answer: Yes, the protocol specification states that the provider responds to an invalid request with a fault that should contain the (domain-specific) terms which the provider could not accept. The fault also contains the negotiability constraints on the terms.

3.2.3.2. Specifying priorities

Question: Is there a way to specify priorities between requirements expressed in the agreement?

¹ In the current (March 2004) draft of the WS-Agreement specification, symmetry for agreement creation and/or negotiation has been defined. The initiator can expose itself to the scheduler via the same Web service interfaces, which enable the scheduler to send messages to the initiator.

Answer: Yes. The WS-Agreement term meta-language is based on WS-Policy [WS-Policy, 2000] which provides XML operators called "compositors" which allow for grouping of terms using OR/XOR/AND logic. In addition, it is possible to flag each grouped term with a "rank" in the group, and therefore specify priorities between terms. There may be needs for more sophisticated ways of specifying priorities. Requirements should be forwarded to the GRAAP-WG mailing-list.

3.2.3.3. OGSi Requirement

Question: Does WS-Agreement require OGSi?

Answer: Yes². WS-Agreement is built on top of OGSi [Tuecke et al., 2003] because the problem space is OGSi-based grids, and because OGSi provides useful standard mechanisms that are not provided by current Web Services specifications. These mechanisms include monitoring of service state, specialization of port types through GWSDL, etc. It can be useful to some agreement initiators to monitor agreement services and/or agreement providers.

3.2.3.4. Roles of the Agreement Service

Several people expressed concerns with the way an Agreement Service merges two notions:

1. The agreement as the state of the negotiation for a service.
2. The virtualization of the provided service.

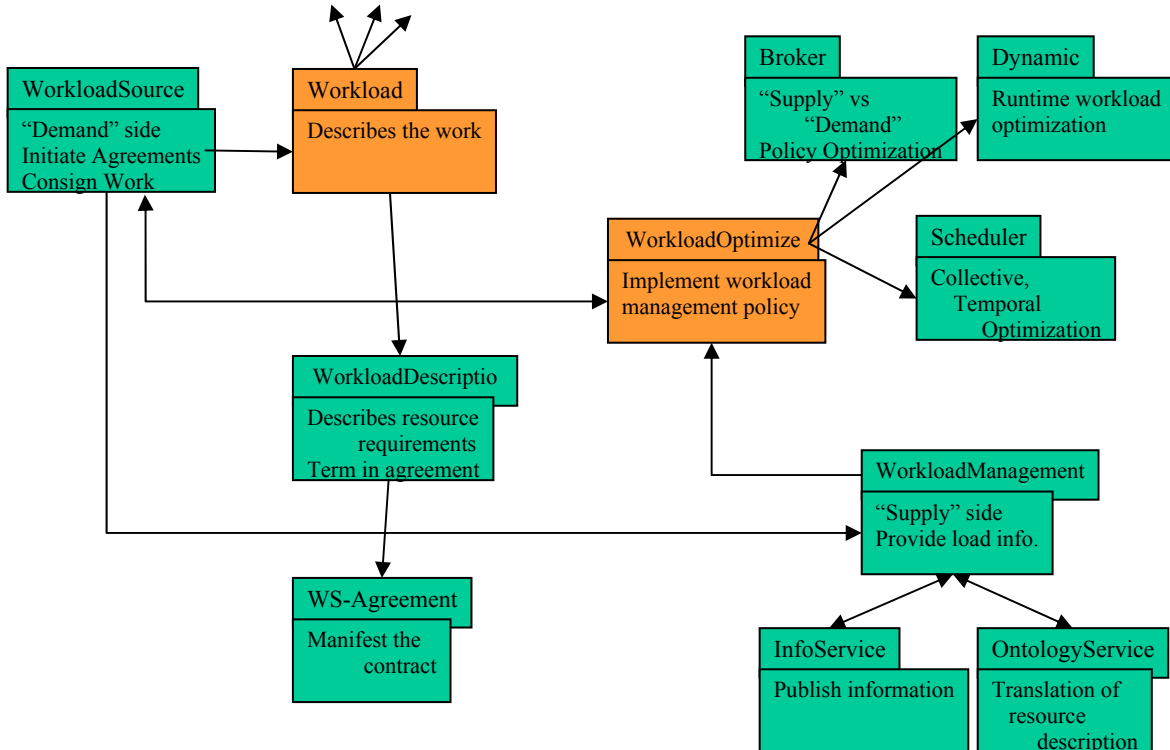
They would have been more comfortable with a separation of those two concepts into two distinct services. However, let us comment that such an approach would increase the complexity of interfacing clients with the scheduling system. Also, an agreement is bound to exactly one service and a service is bound to exactly one agreement, since the service represents the provisioning of the exact terms of the agreement. Therefore, it is hard to see what advantage there would be in decoupling the two notions, and it is more natural to encapsulate the service and its terms into one Grid service instance. In addition, in a service level agreement view of things, part of monitoring an ongoing agreement is monitoring its delivery, so service delivery and agreement terms are tightly bound.

Note: In general, concerns should be addressed by discussions on the GRAAP-WG mailing-list.

3.3. Broker Architecture Decomposition and Interaction Model

We discussed mostly roles and interactions in the diagram entitled "Class Relationships" that David Snelling presented as part of his talk (5.4).

² This answer described the situation at the time of the workshop. Since that date, OGSi has been replaced by the WSRF proposal. It is likely that the WS-Agreement specification will require WSRF, for the same reasons that it previously required OGSi.



Note: further discussion is probably needed in the GGF working groups [OGSA-WG, GRAAP-WG] as the discussion over the distribution of responsibilities in the diagram could have continued past the ending time of the meeting.

3.3.1. Separation between Workload Description and Agreement

One long discussion we had was about the separation of the Workload Description and the agreement. In the model, the Workload Description describes resource requirements. The issue was: why separate the resource requirements from the agreement? Shouldn't resource requirements be an integral part of the agreement, as terms of agreement that initiator and provider negotiate and eventually agree on?

David Snelling mentioned that in schedulers such as Unicore [UNICORE], there is a clear division between resource terms and pure job aspect, and this has been working well in practice. The discussion eventually generalized itself into the issue of separating agreement terms into different logical categories, as the JSDL working group [JSDL-WG] intends to do with the job attributes (note: the JSDL working group has not decided to adopt WS-Agreement yet).

However there are arguments against a separation of agreement terms into different categories. For instance Steven Newhouse raised the issue of how to merge aspects of a job request that are very different in nature, such as the agreed cost of service, the service functionality, etc. Categories are always chosen in an arbitrary manner, and if they are imposed, we end up with a single agreement with multiple components, where cross-cutting aspects such as cost cannot be easily expressed. The work done on WS-Agreement by the GRAAP working group introduces this recurring idea of mixing

and matching terms because one given provider will not necessarily support or require them all. It becomes then a reasonable to define a domain-specific term language that lists possible terms at the same level, and let providers pick and choose which terms they want to advertise as supported (or required) when providing a service.

The question was asked if a scheduler exposed as a WS-Agreement agreement provider can specify what terms it requires: the answer is that not only it is possible, but it should be done by the scheduler. In order to facilitate negotiation, the scheduler should precisely advertise the terms it requires for negotiating a service.

The respective positions being firm, no consensus was reached on the categorization or non-categorization of terms here.

3.3.2. Distribution of “Workload Optimization” and Brokering Roles

The diagram shows some key actors:

Workload Source: makes a request for workload by negotiating with the workload optimizer using the WS-Agreement protocol. Corresponds to the initiator in WS-Agreement.

Workload Optimizer: implements management policies. Acts as a broker to resource managers. Brokers the service provided by the resource to the Workload Source which then interacts with the service representing the resource. Corresponds to a “Schprokerer” as explained in Dave Snelling’s presentation or a “Community Scheduler” as was exposed in the Globus presentation (5.2.5).

Workload Management/Manager: This is the supply side of the equation. It represents a managed resource available (or not) for usage. The Workload Manager publishes information about the resource. It is a resource manager like the Master Managed Job Factory Service (MMJFS) in GT3 [GT3 GRAM, 2003].

A question was raised: can we do symmetry here as well, i.e. can a broker, not a mere scheduler or resource manager, ask for jobs if resources are idle? The answer is that if we ensure symmetry in WS-Agreement, we should be able to ensure it easily here as well. The broker is an agreement provider as well and uses the WS-Agreement protocol to communicate with its clients as well as the end resource managers. However how does the broker coordinate resources it talks to and ensure optimized brokering? In Condor, the matchmaker sends a hint to the resource explaining it was matched and thus will not be considered for matching for some duration. A claiming protocol is then used by the scheduler to claim resource for usage (the claim is verified because claim information could be out of date). A mechanism inspired by the Condor matchmaker could be used in the Broker Architecture.

3.4. Hierarchy of brokers

At the architectural level, a hierarchical structure between brokers/community schedulers is assumed (although it is not represented on the diagram). In fact, a broker acts as a user when negotiating with another broker.

In the hierarchy, the broker represents what is underneath it (i.e. the managed resources it brokers), therefore provides a way to scope policies such as optimization goals.

Notes:

Certain users/schedulers can access resource managers directly, not necessarily through brokers only.

A broker does not necessarily have exclusive access to the resources it uses, and can share resources with other brokers.

3.5. Conclusion

It is encouraging to notice a convergence of concepts exposed during this workshop by different people toward:

1) Matching scheduler capabilities and job requests using service-level agreements to express jobs as quality of service constraints (ex: acceptable time frame for beginning of a job as opposed to fixed time) as opposed to fixed or “hard” values, so that resource consumers and owners can reconcile their competing interests by finding compromises between the two extremes of advance reservation and pure batch scheduling.

2) Hierarchies of (community) schedulers/brokers that do simple brokering or composition of resource-provided services in order to do optimization of resource usage or synthesis of new services. Brokers coordinate resources according to policies geared toward interest of their scope (optimization at that scope). We move away from optimization at grid level because it would require global, centralized control, which does not exist in a decentralized environment such as the Grid. Instead, we move toward distributed scheduling with local information.

Finally, a word on terminology. These intermediaries between users and service providers / resource owners have been designated during this seminar as well as in previous literature as “brokers”, “meta-schedulers”, “schprokerers”, “community schedulers”, among other terms. If the concept is to be universally used and accepted, it sounds necessary to agree on a precise and unambiguous definition and a commonly used English term.

3.6. References

Andrieux, A., Czajkowski, K., Lam, J., Smith, C. and Xu, M. (2003) *Standard Terms for Specifying Computational Jobs (Proposal)*, Global Grid Forum. Available at http://www.epcc.ed.ac.uk/~ali/WORK/GGF/JSDL-WG/DOCS/WS-Agreement_job_terms_for_JSDL_print.pdf.

Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S. and Xu, M. (2003) *Agreement-based Service Management*. Global Grid Forum. https://forge.gridforum.org/projects/graap-wg/document/WS-Agreement_specification/en/1.

GRAAP-WG: Global Grid Forum. <https://forge.gridforum.org/projects/graap-wg>.

GT3 GRAM: *Resource Management*. (2003) In the Globus Toolkit 3 Web documentation. Globus Alliance. Available at <http://www-unix.globus.org/developer/resource-management.html>.

JSDL-WG: Job Submission Description Language Working Group. Global Grid Forum.
<http://www.epcc.ed.ac.uk/~ali/WORK/GGF/JSDL-WG/>.

OGSA-WG: Open Grid Services Architecture Working Group. Global Grid Forum.
<https://forge.gridforum.org/projects/ogsa-wg>.

Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maquire, T., Sandholm, T., Snelling, D., and Vanderbilt, P. (Editors), (2003). *Open Grid Services Infrastructure (OGSI) Version 1.0*. Global Grid Forum. draft-ggf-ogsi-gridservice-29.

UNICORE: Unicore Forum. <http://www.unicore.org/>

WS-Policy: *Web Services Policy Framework (WS-Policy)* (2002) BEA, IBM, Microsoft, and SAP. Available at <http://msdn.microsoft.com/ws/2002/12/Policy/>.

4. Performance

This discussion aims to be representative of the topics discussed at the *Open Issues in Grid Scheduling* meeting in Edinburgh. It is by no means an exhaustive survey of this work area.

4.1. General Issues

4.1.1. The use of performance data in workload scheduling/brokering

Performance plays a large part in workload scheduling/brokering. Research on Grid middleware has tended to separate out levels of operational functionality – performance information is used with a finer degree of granularity at a local (intra-) domain level (one administrator), and a coarser degree across a wide-area (multi-domain) system (multiple administrators) (see 5.1.3). The former supports the use of performance information at the level of *scheduling* algorithms (for example), where local optima might be sought in the improvement and use of performance metrics (see 5.1.3, 5.2.3, 5.3.2). The latter supports the use of performance information when *brokering* for services, a task that might be facilitated both through supporting standards (see 5.2.4, 5.2.5, and also [GRAAP-WG]) and also through intermediaries such as software agents (see 5.1.3, 5.2.5, 5.3.1).

The scalability of the performance data (which are described in more detail below) and the workload scheduling/brokering algorithms (also described below) that use this data are important: the aim is therefore to move towards systems that offer distributed scheduling with local information, rather than centralised scheduling based on distributed information (see 5.4).

4.1.2. Data-aware scheduling

There are a number of methodological issues that impact on scheduling performance. One such approach is *data-aware scheduling* (see 5.1.2), the process of scheduling the tasks close to the data that they require. Considerable performance improvements are to be gained by taking into account the amount (and speed) of data transfer that is required during a task execution. This functionality is to be found in the JOSH (JOb Scheduling Hierarchy) software [Sun Microsystems, 2004], a new GT3-based hierarchical scheduler for the Sun Grid Engine. This system takes load and data proximity into account. An interesting aspect of the implementation is that it splits a job into three parts (pre-, main and post-staging). The advantage of this is that the pre- and post-data handling can be done using a dedicated data transfer queue, which ensures that computation-based HPC tasks are not blocked.

4.1.3. Workflow

Performance is also affected by the order in which a task is carried out. It may be a single unit of work or perhaps a sequence of ordered events. In the latter case performance may be based on the task composition – the performance of a workflow is based on the sum of its parts – but in practice it is not so straightforward. Predicting workflow requirements is

not easy: the workflow may, for example, change during the execution (because of the result of one of the earlier operations); the workflow may be persistent or it may in fact be dynamic (on demand), and of course the compositional units may be executed in parallel. In each case the user requires assurances about the completion time.

When scheduling this type of problem it is not enough to schedule one work unit and then wait until that is complete before planning the schedule for the next. One solution to avoiding latencies between units is to use *advance reservation* (see below). However, reservation does not favour the resource owner (if a reservation is made and then cancelled at the last minute). Current schedulers offer two options – to run the job when it reaches the head of the queue (batch) or to run the job at a precise time (reservation).

Clearly, the choice of technique will impact on performance. Perhaps an alternative is not to try to optimise for resource usage and throughput, but to consider softer bounds such as the *soonest-start* and *latest-end* for a task (see 5.5). User- and system-level policies (service-level agreements) provide a framework that allow this to take place.

4.1.4. Service Level Agreements

The experience gained from designing traditional scheduling systems (see 5.3.3, 5.2.2, and 5.3.5) confirms that one way of reconciling the competing demands of consumers and providers is to enable them to negotiate service-level agreements (SLA).

An SLA is a *contract* between a resource provider and a client specifying the quality of service (QoS) that can be expected (for example, the time by which a task will be dispatched). These contracts do not (necessarily) express optimal requirements, but negotiating these constraints does enable both parties to reach a mutually satisfactory agreement (see 5.3.3).

One way of enabling SLAs in a multi-level distributed system is to have tiers of policies, which might include: policies on resources; policies on users, and policies on communities (aggregations of applications, resources, or both) (see 5.2.5). Performance measures play a large part in ensuring that these policies are adhered to, both in providing data that can be used to evaluate compliance with these policies and in providing data to support revisions to the service if this is not the case.

Governance of SLAs may be performed by administrators or by autonomic intermediaries (agents, for example), and there are emerging standards to support this SLA-based mode of operation; see WS-Agreement from the GGF GRAAP-WG [GRAAP-WG] and resource ontologies, for example, which wrap domain-specific terms and support extensible monitoring.

4.1.5. Reservation

Advance reservation can be viewed as a pre-agreement, where this agreement contextualises domain terms (such as the amount of RAM available to the task) that must be satisfied at the time/place at which the task is run.

The time at which a reservation (or execution plan) is made opens up a range of possibilities: on the one hand, the binding of tasks to resources can be left until the resource is ready (a *just-in-time* strategy); alternatively this binding could be made in advance but changed at the last minute (a *lazy* strategy); or the reservation could in fact

be made well in advance (an *eager* strategy) (see 5.2.2). Each strategy has its advantages and disadvantages: JIT planning is highly dynamic and allows for resources to come and go; eager planning on the other hand means that it is easier to coordinate resources, although this approach does not necessarily scale.

Historical performance profiles can be used in reservation to predict the execution requirements of tasks. There are inherent problems with this however, one of which is the large number of variables implicit in any one of these profiles (for example, the number of CPUs, the CPU speed, which OS is used, the real memory, the disk speed, the available bandwidth, network latency, the number of other users etc.). Other assumptions, such as the price of a resource being constant over time, can also be misleading.

It is argued that planning might alternatively be based on scalable, persistent criteria such as budget and deadline (guaranteed completion time), thus supporting an economic modelling type approach (see 5.1.4) – although this still requires performance measures (such as load predictions) to determine whether an allocation (of task to resource) is viable. Additional techniques for the use and analysis of performance data in this context are still required (see below).

4.2. Performance metrics

4.2.1. Performance data

Scheduling is usually tied to the underlying data that is available. This data might be static (a machine specification, for example) or dynamic (the load on that machine or the queue length), and the value associated with this data may also vary (for example, the cost of a resource may not be constant).

Ensuring the quality of this performance data in large, heterogeneous distributed systems is difficult, as by the time this distributed-state information becomes available it may be out of date (in the case of bandwidth for example). It is also misguided to assume that this data is necessarily (i) available, (ii) accurate, and (iii) remains unchanged (see 5.2.1).

The *quality* of the performance data is therefore paramount and as a result a range of supportive data-collection tools have been developed, including NWS (Wolski), Pinger (Cotrell) and TCP performance monitoring [Downey] for the performance analysis of underlying resources, and PROPHECY [Taylor et al., 2003] and PACE [Nudd et al., 2000] for the modelling of applications. It should be noted in this context that capturing data is only one aspect of the problem. MDS-2 includes data on CPU speed and RAM size etc. but it is not clear, for example, how this necessarily relates to computing *power* (which is not scalar and is often shared).

The *life-time* of this data is also important. This has been demonstrated in the prediction of large file transfers [Vazhkudai and Schopf, 2002; Faerman et al., 1999], where NWS (for example) is shown to provide predictive data on the current state of the system, but not on what the average load is going to be in, say, 20 minutes' time. Prediction should also capture a range of possible values to account for the difference between the actual and predicted data values. Furthermore, it has been shown that scheduling with variance can result in a better mean performance and less variation in the overall execution time

[Yang et al., 2003]. The aim therefore is not prediction per se, but prediction over a particular time period with an integrated model of the possible variance.

4.2.2. Scheduling performance metrics

A number of different scheduling performance metrics underlie current state-of-the-art scheduling systems. These include deadline (see 5.1.3), guaranteed completion time (see 5.1.4), price, average service time (see 5.1.5), start and end time (see 5.3.3), etc.

These terms are not defined here: the reader is referred to the GGF Grid Scheduling Dictionary WG (SD-WG) document on *Grid Scheduling Dictionary of Terms and Keywords* [SD-WG, 2002] and to the selected terminology used in the *Open Issues in Grid Scheduling* presentations themselves.

4.3. Algorithms

4.3.1. Planning vs. scheduling

There is an operational difference between *planning*, which is a client-side operation where the resources cannot be controlled, and *scheduling*, where there is direct control of the ordering of tasks on the compute resources. The algorithms, policies and evaluation (performance) models used by each will affect the underlying performance of the resulting solution.

Similarly, any intermediary (broker) process that performs resource quoting (producer) or resource discovery (consumer), strategy-based selection, and then task assignment (to those resources) will rely on cost-model-based negotiation before selecting/requesting the resources (see 5.1.3, 5.3.1).

A number of algorithms, policies and cost models have been developed for best performance. These occasionally strive for constrained optimal solutions (see 5.3.4), but more often (because of real-time requirements) involve seeking a local minimum. There are a number of good examples of these derived from the world of AI (see 5.1.3, 5.2.3, 5.3.2).

Alternative economic approaches have been proposed (see 5.2.4, 5.1.4), which differ from the optimising-based approaches as they aim to satisfy certain criteria (for example, cost), rather than seeking a solution that offers the best performance. It is also true that achieving optimal operational efficiency is not always appropriate (see 5.3.3), as policy rules and the fact that the current task is not on the 'critical path' may be more important. It is worth noting that in these cases performance might be measured by some other means (such as how satisfied a customer is) rather than approaches based on task deadlines etc.

4.4. References

Downey, A. *TCP Self-Clocking and An empirical model of TCP performance*, <http://allendowney.com/research/tcp>.

Faerman, M., Su, A., Wolski, R. and Berman, F. (1999), *Adaptive Performance Prediction for Distributed Data-intensive Applications*, HPDC-8, Redondo Beach, CA.

GRAAP-WG: Global Grid Forum. <https://forge.gridforum.org/projects/graap-wg>.

Grid Scheduling Dictionary WG (SD-WG) (2002) Grid Scheduling Dictionary of Terms and Keywords, forge.gridforum.org/projects/ggf-editor/document/GFD-I.11/en/1.

Nudd, GR, Kerbyson, DJ, Papaefstathiou, E., Perry, SC., Harper, JS and Wilcox, DV., (2000) *PACE – A Toolset for the Performance Prediction of Parallel and Distributed Systems*, International Journal of High Performance Computing Applications, 14(3):228-251.

Sun Microsystems (2004) JOB Scheduling Hierarchically (JOSH) – Going Global with Globus V3 and Grid Engine, <http://gridengine.sunsource.net/project/gridengine/josh.html>.

Taylor, V., Wu, X and Stevens, R., (2003) *PROPHECY: A Web-based Performance Analysis and Modelling System for Parallel and Distributed Performance*, TOOLS 2003, Illinois.

Vazhkudai S and Schopf, J, (2002) *Predicting Sporadic Grid Data Transfers*, HPDC-11, Edinburgh.

Yang, L., Schopf, J. and Foster, I. (2003), *Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments*, SC'03, Phoenix.

5. Workshop Presentations

This section lists the presentations that were given at the Workshop. Where possible we have included the original abstracts. The slides from the presentations are available on the NeSC website, at <http://www.nesc.ac.uk/action/esi/contribution.cfm?Title=309>.

5.1. Session One: Current Grid Work

5.1.1. Overview of a Grid Scheduling Paradigm

Ali Anjomshoaa, EPCC, University of Edinburgh

Grid Scheduling is very much a part of the management of the Grid fabric. This talk introduces the concept of Grid entities within the Grid fabric, which can be attributed with one or more "roles". These entities need to be managed according to the roles that they possess, much like an employee is managed within a company according to his or her various roles. Different role types are bound to specific "management environments" with their specific sets of management rules and protocols. The interdependencies between these management environments are subject to management according to an interoperability layer that encompasses the commonalities between them. The concept of meta-scheduling is introduced as the process of managing role and Grid entity interdependencies within Grids.

5.1.2. Grid Scheduling Issues in the Sun Data and Compute Grids Project

Terry Sloan and Ratna Abrol, EPCC, University of Edinburgh

The Sun Data and Compute Grids projects aims to develop a compute and data grid scheduler based around Grid Engine, Globus and a variety of technologies. This presentation describes the functional aims of the project and the tools being developed to meet these. The presentation then elaborates on some of the scheduling-related issues tackled during development of these tools.

5.1.3. Performance-Responsive Scheduling for Grid Computing

Stephen Jarvis, University of Warwick

The development of supportive middleware to manage resources and distributed workload across multiple administrative boundaries is of central importance to Grid computing. Active middleware services that perform look-up, match-making, scheduling and staging are being developed that allow users to identify and utilise appropriate resources that provide sustainable system- and user-level qualities of service.

This presentation describes two performance-responsive Grid scheduling services that address the implications of executing a particular workload on a given set of resources. These services are based on an established performance prediction system that is employed at both the local (intra-domain) and global (multi-domain) levels to provide dynamic workload steering. These additional facilities bring about significant performance improvements, the details of which are presented with regard to the user-perceived quality of service and to resource utilisation.

5.1.4. Scheduling Parametric Jobs On The Grid

Jonathan Giddy, University of Cardiff

5.1.5. Dynamic Server Allocation in Heterogeneous Clusters

Jennie Palmer, University of Newcastle

We examine the optimization of a system where the servers in a cluster may be switched dynamically and preemptively from one kind of work to another. The demand consists of two job types joining separate queues, with different arrival and service characteristics, and also different relative importance represented by appropriate holding costs. The switching of a server from queue 1 to queue 2, or vice versa, incurs a cost which may be monetary or may involve a period of unavailability. The optimal switching policy in the case of bounded queues is obtained numerically by solving a dynamic programming equation. Two simple heuristic policies - one static and one dynamic - are evaluated by simulation and are compared to the optimal policy. The dynamic heuristic is shown to perform well over a range of parameters, including changes in demand.

5.2. *Session Two: New Ways to Schedule*

5.2.1. Information And Scheduling: What's Available And How Does It Change

Jennifer Schopf, Argonne National Laboratories

5.2.2. Eager, Lazy and Just-In-Time Planning

Todd Tannenbaum and Alan De Smet, University of Wisconsin

5.2.3. Swarm Intelligence and Static Heterogeneous Multi-Processor Scheduling

John Levine and Graham Ritchie, University of Edinburgh

5.2.4. Trading Grid Services in the UK e-Science Grid

Steven Newhouse, Imperial College London

5.2.5. Agreement-based Distributed Resource Management

Alain Andrieux and Karl Czajkowski, Globus Alliance

Managing resources in the distributed environment of a grid implies coordinating them in a decentralized manner. But this is made difficult by issues in traditional batch scheduling such as the competing goals of resource owners and resource consumers.

In order to deal with such problems we propose an approach based on negotiation of agreements as contracts between providers and consumers. Expressing job definitions using flexible requirements and capabilities enable job requesters and providers to reconcile their conflicting interests through negotiation. Policies proper to each party allow for decentralized management of resources using local rules, while community policies and global optimization goals are enforced by “community schedulers”, that is, facades to sets of resources which can broker existing services and/or synthesize new

ones. A hierarchy of community schedulers linked via requester/provider relationships provides different levels of application of policies as well as the opportunity to decompose the agreement negotiation accordingly.

The WS-Agreement specification, which is still very much a work in progress, was designed so as to enable such an approach to resource management. It leverages standard Grid services mechanisms to provide generic port types (i.e. Web services interfaces) for negotiation of service agreements between requesters and providers. It also seeks to specify a meta-language for expressing flexible service capabilities and requirements. Domain-specific agreement languages and potentially specialized port types are meant to be defined for various service provisioning tasks such as job submission and advance resource reservation.

Following such an open model of negotiation will enable a permanent global Grid to evolve in-place by allowing dynamic incorporation of new types of resources and services as well as policy changes without disruption of the infrastructure.

5.3. Session Three: Insights From Traditional Scheduling

5.3.1. Multi-agent Based Scheduling

Djamila Ouelhadj, University of Nottingham

Autonomous Agents and Multi-agent systems (MAS) represent a new developing area of research. This area of research is now reaching a level of maturity, enough for MAS to be applied as a technology for solving problems in an increasingly wide range of complex applications. Recently, agent technology has been considered as a novel approach for developing complex and dynamic scheduling manufacturing systems. This presentation outlines the drawbacks of static and centralised scheduling systems and gives a review of currently developing research on dynamic scheduling using a distributed multi-agent approach. The motivations, advantages, potentials, design, and applications of this approach for dynamic scheduling are discussed. Finally, a case study on the use of multi-agent systems for integrated dynamic scheduling of steel production is presented.

5.3.2. Evolutionary Approaches to Scheduling

Colin R. Reeves, Coventry University

Evolutionary algorithms (EAs) have become a popular method for solving difficult realworld scheduling problems. In this talk, some fundamental characteristics of EAs will first be described—population-based search, ‘genetic’ operators, and fitness-based selection methods. Then some of the ‘traditional’ scheduling problems to which EAs have been applied, usually characterised as flow-shop, job-shop and open-shop problems, will be discussed. Some variations will also be described, including the need for precedence constraints, intermediate storage, ready times and due dates.

It is stressed that, although often touted as ‘black-box’ algorithms, EAs actually tend to have mediocre performance unless problem-specific information is built into their encodings and operators. Some recent work on fitness landscapes will also be touched upon, and ways in which this work can usefully be embedded in an EA framework will be addressed.

Finally, some of the advantages and disadvantages of using EAs (rather than some other heuristic search method) will be discussed, and some speculations about future developments will be made.

5.3.3. Experience from Designing Transport Scheduling Algorithms

Ray Kwan, University of Leeds

5.3.4. Constraint Programming In Scheduling

Edward Tsang, University of Essex

5.3.5. Industrial Applications of Constraint Based Scheduling

Helmut Simonis, Parc Technologies Ltd

In this talk we discuss some principles of using constraint-based tools for scheduling in an industrial context. We present global constraints, which in recent years have become the main focus for developing powerful propagation methods for solving large scale combinatorial optimisation problems with constraints. For scheduling, we use a combination of temporal and resource constraints, which can for example model disjunctive and cumulative resources, choices between alternatives machines and producer/consumer constraints about intermediate products between multiple production steps.

These constraints can be used together to describe a manufacturing process and, combined with a problem specific search routing, allow the generation of high quality schedules for complex scheduling problems.

In the second part of the talk we describe a number of industrial scheduling solutions that have been developed over the last ten years using this constraint technology. These range from the PLANE system of Dassault Aviation for mid-long term production scheduling, over refinery scheduling systems (FORWARD, Technip) to glass manufacturing (ORDO-VAP, VCA) and short term animal feed production (MOSES, Dalgety). At the end we briefly describe 'Bandwidth on Demand', which provides on-demand, high quality of service bandwidth for specific time periods on an MPLS-based computer network.

5.4. Overview: A Broker/Scheduler Architecture for Grid Services

Dave Snelling, Fujitsu Laboratories of Europe

This talk proposes a basic architecture for the integration of broker and scheduler technology in a Grid Service context. Given basic requirements such as: a scheduler may use the services of a broker, brokers and schedulers may be hierarchically organized, the notion of a commitment from a service is captured by a WS-Agreement, etc. A broker/scheduler agreement language would need to involve a number of states (e.g. Request, Bid, Reservation, Commitment, Spent) and a language to describe a client and service policy. This talk will present a straw-man architecture covering these issues.

5.5. Overview: Service Level Agreement Based Scheduling

Jon Garibaldi, University of Nottingham

Jon MacLaren, University of Manchester

Many Grid Computing Use Cases require complex workflows to be mapped onto distributed computational resources. Naturally, consumers of this technology expect assurances regarding the completion time of such jobs. Unfortunately, the scheduling systems controlling the underlying resources do not provide any information on the expected run time of jobs, unless the run time has been fixed by the use of an advance reservation. Unfortunately, advance reservation is restrictive to the scheduler, and harms machine utilisation, which is often linked to the resource owner's revenue. While sufficient, advance reservation is often not required; some bounds on soonest start and latest end time could be sufficient. Based on these observations, a new UK project is developing alternative scheduling algorithms.

We will capture the acceptable start/end information as part of a Service Level Agreement, one of which will be negotiated for every job in the system. The scheduling problem now starts to look more like a "traditional scheduling" problem, like timetable optimisation or hotel room booking. Strategies such as overbooking become important, etc. The scheduling problem is further complicated by the introduction of dynamic renegotiation, which has arisen as a requirement for RealityGrid, a UK e-Science Pilot Project. During the talk, we will describe the research problem in more depth, and explain how the project team, combining Grid Scheduling experts, and "Traditional Scheduling" experts aims to solve it.