Optimization Strategies for Large-Scale Distributed Computing and Data Management in the Presence of Security and Other Requirements

Gordon Peter Stewart

Submitted in fulfilment of the requirements for
the Degree of Doctor of Philosophy

December 2012

School of Computing Science
College of Science and Engineering
University of Glasgow

Abstract

The growth of e-science and grid computing has presented new opportunities to researchers working on distributed, collaborative projects, or for whom access to large computational and data-storage resources is essential. There is, however, much work to be done to improve the experience for these users, in particular by mitigating the effects that failing to consider fully the requirements of their processing tasks (or jobs) can have on the likelihood of success. Any improvement in this regard would be significant, not only because of the beneficial effect it would have on useful resource utilization, but also due to the improved perception of grid computing it would engender among its users.

Allocating jobs to resources is never easy, and this is particularly true when the resources are of heterogeneous construction, geographically distant, and owned and operated by a variety of institutions. Attempts to allocate jobs intelligently must consider not only the strict hardware requirements of the job and whether these match a particular candidate resource, but must also consider requirements related to the security infrastructures in place (including, for example, the rights of the user who wishes to run the job, the licences associated with the application to be run and the data it will use and produce), how the job might affect other users of the resource, and indeed how the behaviour of other users of the resource might affect the job.

This thesis examines the data, security and legal requirements of typical collaborative research projects, and discusses how these requirements suggest particular constraints that will govern any jobs which need to be executed. It reviews some of the technologies which are common in this field, and describes current state-of-the-art technology in this area. This thesis then presents a framework for describing requirements, along with an algorithmic method for allocating jobs to resources. Case studies and an analysis of performance are presented using these algorithms, which show how they build upon, and improve, the state-of-the-art in this domain.

Contents

List of Tables

List of Code Excerpts

List of Figures

Thesis Statement

The author asserts that the current generation of grid schedulers and resource managers fail to consider fully the diverse range of requirements that have some bearing on the success of any submitted job. These include the requirements of other, competing, uses of a resource, the location of required data, the authorization-related rights of a user, and the licences of the applications and data involved.

It is feasible to express these requirements prior to job submission, and to extract information from the execution resource necessary to determine the likelihood that such requirements can be satisfied or otherwise managed, in order that the allocation of jobs to resources—and, ultimately, the experience of users—may be improved.

This research investigates and proposes methods by which the above can be achieved, and seeks to prove the effectiveness of these proposals by implementing a modified resource brokering system which takes account of this information. This has been incorporated within a test infrastructure, in order that effective comparisons can be made between existing and proposed techniques.

Acknowledgements

I would like to thank those involved in the supervision of this thesis, namely my first supervisors Professor Richard Sinnott and Doctor Wim Vanderbauwhede, and my second supervisors Doctor Peter Dickman and Doctor John Watt.

I would like to thank Professor Asen Asenov who provided funds to allow some of the latter activities from this work to be completed. Furthermore, thanks are due to both Richard and Asen for allowing me the time to work on this Ph.D. despite having a full-time 'day job' throughout.

This work was influenced greatly by the EPSRC-funded project *Meeting the Design Challenges of Nano-CMOS Electronics* (EPSRC reference EP/E003125/1) and I would like to offer my thanks to all those, too numerous to mention, who contributed to that project. Particular credit is due to those who used, broke and provided feedback on a variety of off-the-shelf and bespoke job management tools and applications.

I would like to thank friends and colleagues at the University of Glasgow from both the National e-Science Centre, where this work was begun, and the School of Engineering, where it was concluded. Special thanks are due to Doctor Dave Reid and Doctor Gareth Roy for their useful advice on the subject of statistics and the use of R, as well as to all who provided many valuable nuggets of advice on the topic of thesis-writing, and particularly to Doctor Jon Trinder for sharing the joys of working on a doctorate on a part-time basis. Lastly, I'd like to offer my appreciation for the meticulously planned, concerted nagging campaign which ensured that this thesis did eventually materialize; if I never again hear the question "Have you finished it yet?", I shall be very happy.

Finally, I would like to thank my family for their support, as always.

Gordon Stewart
University of Glasgow
December 2012

Declaration

I declare that, except where explicit reference is made to the contribution of others, this dissertation is the result of my own work, and has not been submitted for any other degree at the University of Glasgow or at any other institution.

Gordon Peter Stewart
14th December 2012

Chapter 1
Introduction

This chapter introduces the grid computing paradigm, and describes the difficulties involved in scheduling large numbers of jobs across a series of heterogeneous resources. It then proceeds to introduce the work which will be described in the remainder of this thesis.

1.1 Overview

Many scientific and engineering investigations today involve significant computational effort. This work will often be divided into logical units—units which shall hereafter be referred to as *jobs*—and consigned to a computational resource—a supercomputer or cluster—for execution. Often, a user will have the choice of more than one suitable resource to which to send their jobs; for example, an academic researcher in the United Kingdom will likely have access to resources provided by the National Grid Service (NGS)[1], as well as clusters maintained locally at a departmental or institutional level. The presence of more than one such resource to which a job could potentially be submitted introduces an important question: how can the most suitable resource for a particular job be chosen?

Conventional job scheduling, of the sort most often employed on computation clusters, is typically based on maximizing the throughput of jobs and the overall utilization of local resources [217]. Due to the need to build generic schedulers, it is of little surprise that rarely is much consideration given to requirements specific to applications, other than some simplistic categorization of the job by the user depending upon its anticipated duration (i.e. will it be short, medium or long, according to some predetermined definition) and an implicit assumption that if the necessary requirements were not in place on the

[1] http://www.ngs.ac.uk

target resource, the user would not have submitted the job in the first place. Indeed, if all an application does is print an error message and then exit, most schedulers will consider the job to have completed successfully. This, coupled with the complex software environment in existence on many grid resources, means the process of identifying and tracking failed jobs can quickly become incredibly problematic [169, 170].

However, by taking full advantage of all information concerning the characteristics (i.e. the requirements and constraints) of a job and the environment available prior to submission—information that can be gleaned from resource requests in submission documents, well-chosen meta-data, and by monitoring system state—the ratio of successful to failed jobs can be greatly improved.

Users from high-performance computing (HPC) or high-throughput computing backgrounds will want to take advantage of the many resources available for academic use, but will naturally also want to integrate any local HPC resources to which they may have access. In a perfect world, the choice of resource would be made automatically and, from the user's point-of-view, transparently. In order to do this, a variety of information about each resource is required. This information can be split into two types: that which is largely static—such as the number of processors or processor cores available, the amount of memory per core, the architecture of the processors, etc.—and that which is dynamic—for example, the number of running or queued jobs—and so must be updated regularly. This information must be matched against the specification of each job or collection of jobs the user wishes to run, in order to suitably match jobs to resources.

It is not sufficient to simply consider the time it would take a particular resource to execute a particular job; any overheads inherent in the choice of resource must also be taken into account. Thus, estimating the time spent copying data from one location to another, or the time it is anticipated jobs will have to wait in the queue prior to being allocated a slot, will be as important as evaluating the raw performance of a cluster.

In order to make realistic calculations based on the need to transfer data from place to place, it is necessary to have a good idea of exactly where the requisite data is located. This will often not be immediately apparent to users—for example, if the data is published on a web server, or made available through a distributed file system—but nevertheless must still be considered. In those situations where the data will be accessed directly when it is required—as opposed to being transferred prior to the start of the job—consideration must be given to the ability of the infrastructure to stream the data as and when necessary.

The legal implications of running a job must also be considered, first and foremost because the licence agreements by which software or other data may be bound might significantly limit the number of candidate resources for the execution of a particular job. It is also folly to submit a number of jobs, all seeking to acquire licences for some commercial software, if the number of jobs exceeds the number of licences available to a particular resource. If too many of these jobs are permitted to run at the same time, a portion will immediately fail having lost out in the competition for software licences; although this has no litigious aspect, and is therefore a lesser risk than running software or data on resources where it is unlicensed, it can be extremely frustrating for users.

The security regime that exists on each target resource must be examined in order to ensure that the user has provided the credentials, and that he or she is in possession of the rights, necessary to run the job. Ideally, an assessment of all aspects of security must be made prior to submission of the job, as being permitted to submit a job to a particular resource does not necessarily imply that permission will be granted for access to any other resources on which the job may depend. Action must also be taken to prevent the situation where a job fails during execution due to a time-restricted credential or permission expiring.

The various issues identified above have not been selected merely at the whim of the author, but have been garnered from his experience working on the EPSRC-funded e-Science pilot project *Meeting the Design Challenges of Nano-CMOS Electronics* [19, 186], a typical large-scale distributed research project.

1.2 Grid Computing

The belief that, at some point in the future, computing resources will be available on demand in much the same way as electricity or water are made available has been around for some time; indeed, in 1961 during a speech at the MIT Centennial, John McCarthy—who among many other achievements invented the Lisp programming language [138]—remarked that "…computing may someday be organized as a public utility just as the telephone system is a public utility…" [84]. The field of grid computing developed from the work done in the late 1980s in what was then termed *metacomputing* [187]; metacomputing described the effort by the NCSA to simplify access to distributed resources—in particular, to the various supercomputers which then existed—and to provide ways to allow these resources to work together on a single problem. The need for improved in-

frastructure and the agreement of standards in areas such as security and accounting were also identified as being of great importance. This theme was developed by the I-WAY project, demonstrated at Supercomputing '95, which among other things developed a prototype software environment to enable easy use of distributed resources [54]. One of the principal researchers from the I-WAY project went on to work on the Globus Toolkit—described originally as a 'metacomputing toolkit' [75]—which has continued to advance the state of software in this area.

Grid computing projects, like their metacomputing predecessors, are typified by several criteria: the involvement of distributed resources not subject to centralized control, the use and application—wherever possible—of open standards, and an effort to provide some non-trivial quality of service [72]. Of course, in order to encourage scientists to engage fully with a grid computing project, it is essential—as has been shown in previous work in this area [76]—that the reliability of any infrastructure developed be maintained, lest users be put off by frequent failure. When executed well, grid computing projects offer the possibility of improved utilization of staff and resources, which in turn can lead to increased productivity and an improved return-on-investment [178]. For these reasons, developments in grid computing, and in distributed computing more generally, are of interest to those working in both the commercial and academic sectors.

1.2.1 Requirements Peculiar to Grid Computing

One of the distinctive characteristics exhibited by almost all projects within the field of grid computing is the need to work with various resources from many different providers, and hence the need to deal with multiple disparate security mechanisms [72, 213]. It is necessary to ensure that the overall view of each user's privileges is correctly and consistently enforced by these different mechanisms, particularly when, as is common in grid systems, a user is mapped to a local account by some identity-mapping scheme [107].

Of particular concern when considering the licence agreements binding software and other intellectual property, is the fact that often the fundamental purpose of adopting grid methodologies is to promote and enhance collaboration among a group of researchers who wish to work more closely with one another. One of the most useful strategies in this regard is to simplify—or, where it was not previously available, to provide—remote access to locally-held resources, be these in the form of computation clusters, software or data. Where access to these resources is bound by licence, it is essential that any grid framework developed respects this. For example, it will often be technically *possible* to

provide a means by which users from one institution can readily gain access to software installed on machines in another, but that does not necessarily make it *legal*.

Another significant issue in collaborative environments is the necessity of being able to ascertain the provenance of data. Being able to answer questions such as "to whom does this belong?" and "on what previous work, if any, was this based?" is not only necessary when ensuring the reproducibility of results—a tenet of scientific research—but is also of vital importance when determining the ownership of the intellectual property and any associated copyright [100]. To further complicate matters, the inter-jurisdictional collaboration that is present in many grid projects means that there may be several, sometimes contradictory, sets of laws governing the use of data, and it is not necessarily clear which set applies in any given case [33]; thankfully, intellectual property law is a reserved matter within the UK (at the time of writing, in any case), and so the position in the author's institution will accord with that in other institutions across the country [118].

1.3 Research Hypothesis

This thesis will show that the effective submission of jobs to distributed computational resources is affected by the requirements and constraints that are imposed upon typical computationally-intensive research projects, and that by giving due regard to these requirements it is possible to devise a schedule that improves the performance of jobs from the perspective of the user, and therefore ultimately impacts positively on the user experience.

The range of potential optimizations is plentiful, with improvements to be made not only to the process of scheduling directly—such as estimation of the holding time for new jobs, and the planning of work with regards to the availability of licences—but also to related aspects that have some bearing, such as the location of data and the presence of security restrictions. Some of these potential improvements will have an impact on the duration of each job, while others will improve the overall success rate, thereby reducing the need to rerun failed jobs; for example, it ought to be possible to prevent jobs being assigned to resources for which there is no possibility of enough licences being made available, or where the prevailing security conditions prevent a job being satisfactorily protected despite all other requirements being met.

The development and implementation of a requirement-oriented framework will allow a range of these requirements to be investigated, and will be used to show how combina-

tions of requirements can affect the process by which an improved schedule is produced. This framework will be evaluated in the context of various real research applications in order to ascertain its suitability in a variety of situations.

This thesis will demonstrate how these contributions to the process of allocating jobs to execution resources improve current state-of-the-art methods, and as a result offer an improved experience for researchers working in this field.

1.4 Publications

The following publication is based on work from this thesis:

> G.P. Stewart & W. Vanderbauwhede, **Improving User Experience of Submitting Jobs to HPC Resources**, Proceedings of the 2012 International Conference on High Performance Computing & Simulation, pp. 635 – 641, July 2012.

The following publications have informed the work in this thesis:

> J.A. Walker, R.O. Sinnott, J.A. Hilder, G.P. Stewart & A.M. Tyrrell, **Optimised Generation of Electronic Standard Cell Libraries with Variability Tolerance through the nanoCMOS Grid**, Philosophical Transactions of the Royal Society A, Vol. 368 No. 1925, pp. 3967 – 3981, August 2010.
>
> R.O. Sinnott, G.P. Stewart, A. Asenov, C. Millar, D. Reid, G. Roy, S. Roy, C. Davenhall, B. Harbulot & M. Jones, **e-Infrastructure Support for nanoCMOS Device and Circuit Simulation**, Proceedings of the Conference on Parallel and Distributed Computing and Networks, February 2010.
>
> R.O. Sinnott, G.P. Stewart, A. Asenov, C. Millar, D. Reid, G. Roy, S. Roy, C. Davenhall, B. Harbulot & M. Jones, **Multi-level Simulations to Support nanoCMOS Electronics Research**, Proceedings of the ASME International Design Engineering Technical Conferences, August / September 2009.
>
> D.Reid, C. Millar, S. Roy, G. Roy, R.O. Sinnott, G.P. Stewart, G.A. Stewart & A. Asenov, **Enabling Cutting Edge Semiconductor Simulation through Grid Technology**, Philosophical Transactions of the Royal Society A, Vol. 367 No. 1897, pp. 2573 – 2584, June 2009.

R.O. Sinnott, C. Bayliss, T. Doherty, D. Martin, C. Millar, G.P. Stewart & J. Watt, **Integrating Security Solutions to Support nanoCMOS Electronics Research**, Proceedings of the International Symposium on Parallel and Distributed Processing with Applications, December 2008.

R.O. Sinnott, C. Bayliss, C. Davenhall, B. Harbulot, C. Millar, G. Roy, S. Roy, G.P. Stewart, J. Watt & A. Asenov, **Secure, Performance-Oriented Data Management for nanoCMOS Electronics**, Proceedings of the 4th IEEE International Conference on e-Science, December 2008.

D. Reid, C. Millar, S. Roy, R.O. Sinnott, G.P. Stewart, G.A. Stewart & A. Asenov, **Prediction of Random Dopant Induced Threshold Voltage Fluctuations in NanoCMOS Transistors**, Proceedings of Simulation of Semiconductor Processes and Devices, September 2008.

R.O. Sinnott, G.P. Stewart, C. Millar, C. Bayliss, D. Reid, G. Roy, S. Roy & A. Asenov, **Meeting the Design Challenges of nanoCMOS Electronics through Secure, Large-scale Simulation and Data Management**, Proceedings of the EGEE '08 Conference, September 2008.

R.O. Sinnott, C. Bayliss, D.W. Chadwick, T. Doherty, B. Harbulot, M. Jones, D. Martin, C. Millar, G. Roy, S. Roy, G.P. Stewart, L. Su, J. Watt & A. Asenov, **Scalable, Security-Oriented Solutions for Nano-CMOS Electronics**, Proceedings of the UK e-Science All Hands Meeting, September 2008.

D. Reid, C. Millar, S. Roy, G. Roy, R.O. Sinnott, G.P. Stewart, G.A. Stewart & A. Asenov, **An Accurate Statistical Analysis of Random Dopant Induced Variability in 140,000 13nm MOSFETs**, Proceedings of the Silicon NanoElectronics Workshop, June 2008.

R.O. Sinnott, T. Doherty, D. Martin, C. Millar, G.P. Stewart & J. Watt, **Supporting Security-Oriented, Collaborative nanoCMOS Electronics Research**, Proceedings of the International Conference on Computational Science, June 2008.

R.O. Sinnott, D.W. Chadwick, T. Doherty, D. Martin, B. Nasser, A.J. Stell, G.P. Stewart, L. Su & J. Watt, **Advanced Security for Virtual Organizations: The Pros and Cons of Centralized vs. Decentralized Security Models**, Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, May 2008.

D. Reid, C. Millar, G. Roy, S. Roy, R.O. Sinnott, G.A. Stewart, G.P. Stewart & A. Asenov, **Supporting Statistical Semiconductor Device Analysis using EGEE and OMII-UK Middleware**, Proceedings of the 3rd EGEE User Forum, February 2008.

L. Han, A. Asenov, D. Berry, C. Millar, G. Roy, S. Roy, R.O. Sinnott & G.P. Stewart, **Towards a Grid-Enabled Simulation Framework for Nano-CMOS Electronics**, Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing, December 2007.

J. Watt, R.O. Sinnott, J. Jiang, T. Doherty, A.J. Stell, D. Martin & G.P. Stewart, **Federated Authentication & Authorisation for e-Science**, Proceedings of the APAC Conference and Exhibition, October 2007.

R.O. Sinnott, A. Asenov, D. Berry, A. Brown, L. Han, C. Millar, G. Roy, S. Roy & G.P. Stewart, **Grid Infrastructures for the Electronics Domain: Requirements and Early Prototypes from an EPSRC Pilot Project**, Proceedings of the UK e-Science All Hands Meeting, September 2007.

1.5 Summary

This chapter has introduced grid computing, and described some of the difficulties and deficiencies that the work in this thesis shall address. Further background, and a review of the state-of-the-art, can be found in *Chapter 2 – Background and State-of-the-Art*. A discussion of the various requirements that will be addressed may be found in *Chapter 3 – Proposed Requirement Framework*, along with some details of additional information needed in order to make decisions about resource selection. Details of the design and implementation of the framework and the allocation strategies that have been developed are discussed in *Chapter 4 – Design of an Improved Resource Broker*. The design and conduct of the evaluation is described in chapter *Chapter 5 – Evaluation*, which includes the particulars of the simulation environment used for this study. Finally, conclusions and closing remarks may be found in *Chapter 6 – Conclusions*.

Chapter 2
Background and State-of-the-Art

This chapter introduces and describes some of the technologies prevalent in the grid computing domain. It also details the state-of-the-art in terms of job scheduling and resource brokering, areas which underpin the research presented in this thesis.

2.1 High-Performance Versus High-Throughput Computing

High-Performance Computing (HPC) is the name given to the class of computational resource designed to offer applications the ability to perform a 'significant' number of operations per second, where the term significant is necessarily relative to the general availability of computational power at the time. High-Throughput Computing (HTC) is the term used to denote those resources able to deliver significant quantities of computational processing capacity over a longer period of time, allowing researchers to address the issue of what can be accomplished over a period of several months or years [26].

In simple terms, the difference can be viewed as follows. In HPC circles, what matters is maximizing the performance of a single job so that it runs in the shortest time possible. As a result, HPC applications often comprise tightly-coupled parallel processes, executing in harmony on a single resource to solve a single problem. On the other hand, HTC is concerned not with how long any one job takes to run, but rather with how long lots of jobs—perhaps tens or hundreds of thousands—take to run. Therefore, the ability to halve the execution time of a particular job by using twice the available resources might be considered a great step forward by an HPC user; however, for an HTC user wishing to run a number of jobs that greatly exceeds the available resource capacity, there would be little advantage, as it effectively allows him to run half the number of jobs (due to the

greater resource requirement of each) in half the time, and so has no overall effect on throughput.

Even when the aim is to maximize throughput, single-thread performance remains an important factor for many applications. The term High-Performance Throughput Computing is occasionally used by those wishing to emphasize this requirement [46]. There is also a developing paradigm where various distinct, HPC-like jobs share data by means of a file system. This necessitates consideration of a combination of HPC and HTC issues, and may be termed Many-Task Computing (MTC) [168].

2.2 Background to Grid Computing

The success of many grid computing projects lies in the process of integrating the various technologies that will be used. In some cases, multiple technologies are brought together in order to pick the best features of each; in others, two or more technologies that largely overlap in terms of their functionality must be supported because of the requirements imposed by different resources which are candidates for use. In order to understand the way in which these technologies may be combined, it is necessary to examine their purpose in a little more detail.

Many grid computing projects adopt a service-oriented architecture [79]. A *service* is an open, self-describing component, offered by a *service provider*, which provides some functionality to a client application. The nature of this functionality can vary considerably, but might include providing methods to query and update a database, or to execute a particular program (a common use of services in the grid domain is to make internal applications externally-accessible over the Internet [124]). Multiple services can be easily composed, with the output from one service being fed into another, and this allows distributed applications to be readily combined in some useful manner [161].

2.2.1 Web Services

A *web service* transfers the notion of a service to the World Wide Web, being a software system identified by a Uniform Resource Identifier (URI) which provides some functionality to a client application [20]. Interaction between client and service traditionally took the form of an exchange of messages using a standard protocol such as SOAP (which is officially no longer an acronym) [92]. Since its introduction by Roy Fielding, an architect of the HTTP protocol [70], the REST (REpresentational State Transfer) ap-

proach [69] has become increasingly common, due to its relative simplicity and reliance on standard HTTP methods (e.g. GET and PUT) [94]. REST ignores several technical challenges—such as reliability and integrity—which are addressed by SOAP, but which are unnecessary for many applications.

It is common for descriptions of web services to be published in a format such as the Web Services Description Language (WSDL) [68] or the Web Application Description Language (WADL) [95], although this is not a strict requirement.

2.2.2 Grid Services

Although plain web services can be used in the grid domain, several research strands have looked at adding additional functionality to these to improve their usefulness.

The Open Grid Services Architecture (OGSA) is the name given to the work by the Globus® Alliance[1] and IBM®[2] to align grid technologies with web technologies, and in particular to the augmentation of web services technology to provide a framework upon which grid services may be constructed [76, 197]. The interface to these grid services is described using an interface definition language such as WSDL, although it should be noted that this describes the form which interaction between client and service will take, rather than the behaviour of the service—i.e. what it actually does—on invocation [79]. The implementation of a service is distinct from its interface; indeed, one service can provide multiple implementations for different platforms. OGSA defines several interfaces which a service may implement, and which are intended to ease interoperability by standardizing the message formats for common operations.

The Open Grid Services Infrastructure (OGSI) [204] provides the foundation upon which OGSA is built. OGSI defines a model that extends conventional web service architectures to support such features as stateful web services, references to service instances and collections of such references. Concerns were raised, however, over the design of OGSI, with complaints suggesting that it was too large, too object-oriented and did not integrate well with existing web services tools [73].

The Web Services Resource Framework (WSRF) [23] was inspired by—and evolved from—OGSI, providing a way of describing the resources manipulated by web services, as well as offering a framework that provides such features as service information collection and aggregation. WSRF is seen as completing the convergence of web and grid

[1] http://www.globus.org
[2] http://www.ibm.com

services [73], making reference when appropriate to other web service standards. For example, the WS-Addressing standard [93], which provides transport-neutral mechanisms to address web services, has been adopted.

WSRF addresses many of the concerns that were raised about OGSI: it divides the requirements into a number of related specifications, allowing a user to select the portions of which he wishes to avail himself. The way in which XML and WSDL are used has been changed in order to make WSRF more suitable for use with other web service tools, and a distinction has also been made between the service (which is stateless) and the resources it accesses (which may or may not be). This was intended to quell the anxieties of the web services community, which viewed OGSI as transforming web services into hugely complex, heavyweight entities [74].

2.2.3 Cloud Services

Cloud computing is a new buzzword favoured by those working in enterprising computing, used to describe distributed-computing systems made accessible by means of the Internet. Although there is presently no consensus on exactly what constitutes "cloud computing" [85], a number of features are typical of the paradigm [134]. Firstly, the infrastructure will be dynamic, and most likely virtualized. Access will be on a self-service basis, so users will be able to make use of resources independently, and the resources themselves will require minimal management. Finally, charges will be levied according to the amount of use a user has made of a resource (i.e. a consumption-based billing model will be adopted). Despite many years of development, there has yet to be a commercially-viable grid computing provider, due in no small part to the complexities surrounding the deployment and management of current grid computing infrastructure [38]; it is this gap in the market that cloud computing providers seek to fill.

Cloud computing has many similarities with grid computing; indeed, one definition of cloud computing suggests that it is simply grid computing given a more user-friendly face [85]. As such, there may be areas where established, open grid technologies and protocols exist which would address problems that clouds have yet to solve, or only solve in a proprietary or overly-specialized manner [80]. It would seem sensible that developers of cloud computing infrastructure attempt to make use of such solutions where possible. For example, one important challenge which is still to be satisfactorily addressed by cloud computing service providers is that of interoperability, with different cloud services making use of an assortment of APIs, many of which are proprietary [56]; this contrasts sharply with the use of open standards preferred by the grid computing community.

One difference between grid and cloud computing is that the resource brokering model will probably look quite different. It is certainly possible that traditional batch queueing systems—such as those described in *Section 2.6.7 – Local Job Scheduling* which are prevalent on today's batch computing resources—will either not be employed, or will only be made available in a heavily-modified form.

2.2.4 The Globus Toolkit

Probably the most widely known of all grid computing software projects is the Globus Toolkit. Globus was originally designed to be a "metacomputing toolkit", allowing its users to create metacomputers (in other words, networked virtual supercomputers) [75]. The strategy employed by its developers was to provide components—covering such issues as resource location, authentication and data-access—from which higher-level services could be developed.

In order to support the collaborative nature of grid projects, the concept of a Virtual Organization (VO) was introduced. A VO encompasses a dynamic, multi-institutional group of individuals and systems, aiming to share access to computers, software, data and other resources [78]. One of the archetypal examples of this sharing is in the tremendous efforts which have been made to provide international access to high-energy physics data from large research centres such as CERN (the European Organization for Nuclear Research) [102]. The identification of particular needs led to the development of OGSA, the Open Grid Services Architecture, which provides standard interfaces and conventions to support the creation and management of grid services; these services are able to make use of existing IT infrastructure, and can be combined as and when required to support the needs of VOs [76]. The Globus Toolkit provides the tools necessary to develop and host these services.

The Globus Toolkit, now in its fifth incarnation, remains under active development, and the latest 5.2 series was released in mid-December 2011. Recent releases have made great strides towards simplifying the installation and configuration process, with many Linux distributions now containing Globus packages in their repositories [87]. Version 5 of the toolkit included a number of enhancements over its predecessors, such as improvements to its Grid Resource Allocation and Management (GRAM) component, which is used to interact with schedulers on execution resources. Progress has also been made towards modernising those components which depend on what are now considered legacy technologies [104].

2.3 Grid Security

2.3.1 Authentication

Authentication is the name given to the process by which the asserted identity of an entity is confirmed [141]; a user is the most common example of such an entity, although the necessity of mutual authentication (i.e. authenticating the application or service with which the user is attempting to communicate) must not be forgotten [165]. In order that this identity can be used in further access control decisions, it is essential that a high degree of trust can be placed in the reliability of the authentication measures; failure to correctly authenticate an entity renders subsequent decisions invalid [101].

There are various methods of authentication available, most of which require the authenticating entity (for simplicity, henceforth referred to as the "user") to prove that it is in possession of some secret information not publicly known. For example, one of the most common authentication approaches adopted in computing requires the user to provide a username (for identification) and a password (for authentication); the fact that the user is able to provide the secret password when prompted gives credence to the fact that the user is who he or she purports to be. This is a commonly-adopted technique outside the realms of computer science, where the presentation of identification and some secret information—for example, a credit card and PIN—is used to confirm identity.

PKI Authentication

A Public Key Infrastructure (PKI) is based around the concept of certificates. A certificate is an object that binds an identity to a public key. Certificates are issued and signed by some central, trusted party; they can then be distributed among participants in a conversation, with each participant being able to verify—i.e. check they were issued by the trusted party—any certificates sent to them. Once a certificate has been verified, the public key it contains may be used to encrypt messages to be sent to its owner, and can also be used to check the signature on messages received from its owner [121].

It is important to note that there are various certificate specifications, such as those provided by Pretty Good Privacy (PGP) [83] and the Simple PKI [62, 63]. To further complicate matters, due to the number of optional components and extensions allowed by some of these schemes, there are often multiple ways of transmitting one type of certificate [10]. The X.509 standard [3] has been adopted by the grid computing community, and hereafter the term "certificate", without further qualification, refers to an X.509-compliant object.

Within the United Kingdom, certificates for use in e-science applications are issued by a certification authority (CA) managed by the National Grid Service. A simple verification procedure, in which the user requesting a certificate identifies him- or herself to a designated member of staff—the registration authority, or RA—at his or her local institution, helps to ensure that certificates are only issued to those with a legitimate need for one. The identity and method of identification is logged by the RA, who must also ensure that the distinguished name of the certificate to be issued will be unique [112]; it is necessary to trust that the RAs will follow these procedures correctly, as the quality of this process affects the level of trust that can be placed in the issued certificates [108].

Issues of trust within a PKI-based infrastructure can be managed by examination of certificate chains: when a new user receives a certificate, it is signed by the issuing certification authority (in this case, the NGS); in turn, this signing certificate is itself signed by another certificate—referred to as the root certificate—held by the same party. By opting to trust the signing certificate, it is possible to automatically trust all its child certificates (i.e. all the certificates which it has been used to sign) without the need to trust each explicitly. This simplifies the process by which access to a resource may be provided to a large community of users.

It is often desirable within a grid computing environment to allow a user to delegate some of his or her privileges to another entity, in order, for example, to allow a job to run unattended on a remote resource. To permit this, an extension to the original X.509 standard was developed which allows the creation of a *proxy certificate* [205], which enables an entity to gain access to the privileges—or a subset thereof—held by a particular user, in the same manner as it would if the original certificate were present [212]. While user certificates are often valid for periods of a year or more, proxy certificates will typically be valid for no more than a week; this provides a degree of damage-limitation in the event that the proxy is compromised.

With the reliance on proxy credentials for authentication, and the distributed nature of the grid computing domain, it is often necessary for several entities to obtain a copy of a proxy credential in order to run a single job. The MyProxy Credential Management Service[3] was designed to simplify access to proxy credentials, storing them in a secure manner until they are required, and then providing various ways in which they can be retrieved. Most importantly of all, users retain full control of their proxy credentials within a MyProxy server, and can place restrictions on the way in which credentials may

[3] http://grid.ncsa.uiuc.edu/myproxy

be obtained [153]. Communication with a MyProxy server utilizes transport layer security in order to protect the credentials during transmission [24].

Shibboleth®

Shibboleth[4] provides a framework for authentication and authorization of web-based resources, and forms part of the Internet2 project[5]. It has recently become prevalent through its use by JISC[6] to provide members of academic institutions in the United Kingdom with federated access to resources [5]. When a user attempts to open a Shibboleth-protected resource in their web browser, communication is initially redirected to a page—known as the "Where Are You From?" page—where the user is prompted to nominate his or her home institution. The user then proceeds to authenticate with this institution and, on successful completion of this stage, is returned to the resource which was originally requested, but with an additional handle that can be used by the resource to make authorization decisions [150, 184]. This handle takes the form of a signed Security Assertion Mark-up Language (SAML) [39] message. Two types of SAML statement are used within Shibboleth: authentication statements, which make assertions about the principal's identity and mode of authentication, and attribute statements, which contain additional information for use in authorization decisions [177].

Authentication based on Shibboleth has several advantages over purely PKI-based methods. Of principal importance is the federated (i.e. decentralized) nature of Shibboleth: records of users are maintained at the institutions to which they belong, and are therefore far more likely to be kept up-to-date when users come and go.

2.3.2 Authorization

Authorization is the process by which the rights and permissions of an entity are determined [141]. For example, when a user wishes to make changes to a particular file, most operating systems will first examine the permissions on the file and its parent directory before deciding whether the user is entitled to make such changes. In order for accurate authorization decisions to be taken, the entity must have been authenticated correctly. If a distinction is made between "identity" trust and "behavioural" trust, establishing identity trust can be seen as the role of authentication, while the notion of behavioural trust can be used as the basis of authorization decisions [21].

[4] http://shibboleth.internet2.edu
[5] http://www.internet2.edu
[6] http://www.jisc.ac.uk

PKI Authorization

There are several limitations to infrastructures based solely on certificates, particularly when authorization is considered. The simplest model, in which the DNs of users are associated with UNIX accounts on the local machine by means of a mapping in a text file [88], does not provide any means to limit the actions of a particular user, beyond simply denying him or her access at all [185]. Furthermore, maintaining a list with an entry for each potential user rapidly becomes an unmanageable task, and also fails to model those situations where it might be desirable to support varying degrees of access for different users (or, indeed, for the same user in different circumstances). In these situations, a more sophisticated infrastructure providing Role-Based Access Control (RBAC) is often more suitable [191]. However, regardless of these failings, this approach is commonly used on many grid resources [181].

Shibboleth Authorization

Although Shibboleth was designed with the principal purpose of addressing matters concerning authentication, it also provides optional support for authorization. Shibboleth-protected resources can conduct authorization checks by examining the assertions provided about a user, and making decisions based on these assertions. For example, it might be the case that one of the attributes includes information on a user's roles within a project, and that only users with a certain role are allowed access to a particular resource. Further limitations—such as a maximum number of concurrent users—may also be enforced, meaning that simply possessing the necessary attributes may not be sufficient to gain access [40]. The attributes that are deemed acceptable by a resource can be further restricted by "scoping" [211], a technique which allows resources to accept or refuse attributes according to the institution from which they originate. Attributes from multiple identity providers can also be linked together, allowing a full picture of a user's privileges to be obtained by evaluating information from multiple sources [210].

VOMS

As previously discussed, the collaborative nature of many grid computing projects means it is common for participants to form a virtual organization (VO). The VO is effectively a management structure, but grouping members according to their affiliation to a certain project, rather than to a particular institution. For example, researchers working on the project *Meeting the Design Challenges of Nano-CMOS Electronics* were awarded membership of the `nanocmos.ac.uk` VO. Within the VO, participants—who may or may not otherwise have a relationship—agree to share resources, and this sharing typi-

cally extends beyond mere file repositories, for example by including access to software or computational resources. The notion of the VO is important because it allows a degree of control to be exerted over otherwise disparate individuals [78].

The Virtual Organization Membership Service (VOMS)[7] is based upon a model in which a VO maintains a centralized service—essentially the front-end to a database of some description—providing information about a user's role within the organization; in many ways, the VOMS server has a status akin to that of the CA in a purely PKI-based infrastructure [32]. This information—which is provided in the form of a signed X.509 attribute certificate [13, 66]—can be accessed by resources for use in authorization decisions, with the effect that the final decision as to whether or not an action ought to be permitted resides with the resource provider [12]. The adoption of a centralized model allows roles to be defined globally within the VO, and negates the need for individual partners to reach agreements with each other, without depriving them of the ability to control access to their own resources. There is also no need to accept and aggregate role information from multiple disparate sources. As ever, there are also problems with this model, including the potential vulnerability of a single, central management service, and also the sheer amount of work required to administer a VO with many users [183].

VOMS supports both push and pull models of attribute dissemination. In the pull model, each resource periodically establishes a secure channel to the VOMS server, and retrieves a list of user DNs matching certain criteria (the criteria on which the resource wishes to base access). In the push model, users first contact the VOMS server to retrieve the appropriate signed assertions which they present to the resource they wish to use [133].

PERMIS

PERMIS[8] is an X.509 role-based privilege management infrastructure (PMI). As with VOMS, additional information about a user is provided by means of an X.509 attribute certificate, issued by an attribute authority. PERMIS may be used in a number of ways, but is most commonly used to provide access control based on roles. In this model, the roles which a user holds are listed in the attribute certificate; resources can check the roles against an access policy to determine whether or not a request should be granted. This system can be enhanced to support hierarchical RBAC—where superior roles inherit the permissions of their subordinates—and constrained RBAC—where further constraints,

[7] http://vdt.cs.wisc.edu/components/voms.html
[8] http://sec.cs.kent.ac.uk/permis

such as time limits, are placed on the validity of roles—as well as a consolidated form, incorporating features from both the hierarchical and constrained models [43, 44].

Alternatives to VOMS and PERMIS

In addition to the technologies mentioned above, there are several other authorization systems in use and a myriad more have been proposed. Of these, CAS[9]—the Community Authorization Service [164]—and Akenti[10] [200] are among the most common.

CAS differs from VOMS in the manner in which the rights a particular entity has are determined: in both cases, group membership and role assignment is handled centrally, but CAS also centralizes handling of rights, while VOMS leaves the interpretation of the membership information to individual resources. PERMIS and Akenti possess a similar architecture to CAS, but are perhaps more feature-rich (and, consequently, are both more complex pieces of software) [163]. There are also striking differences between PERMIS and Akenti, the most notable being in the way security policies are defined: Akenti features hierarchical and distributed policy and use-condition certificates, and concentrates on traditional ACLs; PERMIS stores policies in a single attribute certificate, and supports a role-based access control model [42].

2.3.3 Intellectual Property and Digital Rights Management

In the twenty-year period from 1984 to 2004, the combined value of the net assets of the top ten companies listed on the London Stock Exchange doubled, while the combined value of the companies themselves increased nearly ten-fold. This apparent disparity between the value of concrete assets and the value of the companies is accounted for by the rapid increase in the value inherent in their reputations and, in particular, in their "knowledge capital". It is therefore unsurprising that, in recent years, various parties have taken great interest in attempting to protect this intellectual property [22, 91].

Much work has been done on the topic of Digital Rights Management (DRM), by which we refer to technologies designed to monitor or enforce end-user compliance with the licences of digital property (opinions as to precisely what constitutes DRM vary: some simply define DRM to be "the secure exchange of intellectual property" [160], while a better definition would perhaps be "the management of intellectual property rights for digital content through digital means" [137]). Most of this work has concentrated on the protection of multimedia content—photographs and audio recordings, for example—in

[9] http://www.globus.org/grid_software/security/cas.php
[10] http://acs.lbl.gov/Akenti

order to allow its controlled distribution over telecommunications channels. It is important to remember that software licensing can be considered a special case of DRM, in which the intellectual property to be protected is the software in question.

When consideration is given to the nature of the intellectual property to be protected within typical scientific research—and, in particular, when consideration is given to the detrimental effects of any infringement—it rapidly becomes clear that the overwhelming majority of existing work in the field of DRM does not fulfil the needs of these projects. It is, however, certainly not the case that an examination of previous work is without merit. Although the circumstances may be different, the requirements identified by other investigators may well be valid: for example, it is important that any solution proposed be technically feasible, efficient, flexible and avoid derogation from accepted principles regarding privacy [55].

Licence Management

Licence management is a complex issue, and poses various legal and technical challenges. As has already been mentioned, it is important to respect the terms of the licence agreement, regardless of whether operation of the software other than according to such terms is actively inhibited (for example, by licence management software). Furthermore, it must be borne in mind that the licence managers used by particular pieces of software may make certain types of operation impossible, even if such operation does not contravene the licence. For example, it may be the case that copies of a particular application can only be licensed on machines which are in possession of an individual IP address visible to the machine hosting the licence manager; as this is not the situation on many clusters—where individual cluster nodes typically have IP addresses in a private sub-net, and communicate externally through NAT—it may be extremely difficult to run licensed software in these environments.

Licence management systems designed specifically for distributed computing environments have been proposed. For example, GenLM [51] is based on the concept that a licence may be attached to a particular set of input data, as opposed to granting it to an institution, resource or user. This has the advantage that it no longer matters where the job is run, but requires application vendors to adopt a new way of thinking about the licensing process, allowing as it does users to purchase licences for particular simulations. The Licence Management Architecture [57] similarly provides support for pay-as-you-go licensing models, while attempting to provide some means to support legacy licence managers in a distributed computing scenario.

Protection of Data

Attempts can be made to protect data in a number of ways, many of which have been used, or are based upon schemes used, for the distribution of digital content. It is important to note that many of the schemes *do not* actually prevent the unauthorized copying of data, but rather provide a mechanism by which such copies can be identified. Methods such as digital watermarking and fingerprinting fall into this category, fingerprinting being a form of watermarking in which the watermark contains information which uniquely identifies the copy [115] (a serial number would suffice). In the event of an unauthorized copy being detected, the presence of a watermark within the copy allows the originator to prove that he or she was the creator and is therefore entitled to exercise rights as such; the presence of a fingerprint allows the creator to go further by identifying which authorized copy was duplicated to create the unauthorized copy, and therefore potentially identifying the source of the illegal copy.

A distinction must also be made between functionally-correct watermarking and artefact watermarking. Few people can perceive slight changes in colour or high-frequency sound—hence the success of the compression algorithms used in JPEG [1] and MPEG [2] technology—and for this the latter watermarking technique, which requires that changes be made to the object to be protected, is perfectly acceptable. Such a technique does not work, however, when the underlying data cannot be modified.

Regardless of the technology used, watermarking as a whole is not sufficient to prevent infringement of intellectual property rights, but rather is intended to provide the basis for legal action to be taken [135]. As such, it is of limited use within security-conscious projects and is not worthy of further investigation. Instead, efforts must be concentrated on ensuring that restricted data cannot escape the controlled environment of the grid framework.

2.4 Data Management

Arguably the most important aspect of many research projects relates to the production, management and protection of scientific data. In many disciplines, the majority of this data is file-based, although it is generally recognized that in order to fully understand and explore such data, sufficient and appropriate meta-data must be produced to adequately describe it.

2.4.1 File-Based Data

The Andrew File System (AFS), of which OpenAFS[11] is an open-source distribution, is a popular file-storage medium. Originally developed as part of the Andrew project at Carnegie Mellon University [103], it has been used successfully in large-scale deployments, both as a storage mechanism for user directories [47], and in support of high-performance computing resources [189, 194]. (It must be recognized, however, that AFS may not match the performance of a dedicated high-performance file system such as PVFS [99], although its other benefits will often outweigh such considerations.) In certain situations, AFS provides a richer feature set and better performance than alternatives such as the Storage Resource Broker (SRB) [182]. Of particular note given the variety of platforms encountered when users from different backgrounds and institutions collaborate, OpenAFS clients are freely available for many different operating systems [53].

The AFS file-space is divided into administrative units termed cells [220]. Each cell has one or more file servers allocated to it, and these serve the various volumes that make up the cell's file-space. One of the advantages of AFS with regards to scalability and ease-of-use is that it provides a common name-space, at least across UNIX-like operating systems. On such systems, AFS cells are mounted under the `/afs` directory, with the constituent volumes appearing as subdirectories within the cells; so, for example, the AFS cell operated by the National e-Science Centre is published as `/afs/nesc.gla.ac.uk`, with various project-related cells made available beneath this. Not only does this mean that a particular file can always be accessed by using the same pathname, regardless of the exact combination of operating system and OpenAFS client used, but also that file-servers can be added or removed, and volumes moved from one server to another, without the end-users ever being aware of this; indeed, all aspects of server organization are concealed from non-administrative users. This is an improvement over the situation with systems such as NFS in which the filename given to a particular file will differ from one client machine to another [113], although it is not unique (LegionFS, for example, also features location-independent naming [214]).

The manner in which the file-space can be divided into volumes, and the ease with which these volumes can be replicated, offers various other benefits that can improve performance and reduce the administrative workload. Provided the distribution of files among volumes is logical, it is a simple matter to identify those volumes which contain important data, and to separate them from those which are used simply as temporary or

[11] http://www.openafs.org

scratch directories; the back-up process then becomes a trivial series of volume replications, preferably to a different file-server. It is also possible using the AFS volume replication mechanism to create read-only copies of a volume; hosting such replicas on a server close to where the files will be needed can improve performance in cells which have many geographically-distant members.

Security provision in OpenAFS is based around version 5 of the Kerberos Network Authentication Service [120, 190], originally developed as part of MIT's Project Athena. In addition to authentication, Kerberos provides support for mutual-authentication, allowing clients to ensure that the servers with which they are communicating have not been replaced by nefarious imposters. The traditional Kerberos model, which utilizes conventional password-based authentication, is ideal for interactive use, but is of little help when unattended remote access to AFS resources is required, such as is the case when jobs submitted to a batch submission system reach the front of the queue and begin to execute; in this situation, it is impractical—indeed, often impossible—to require the user to enter his or her password before the job can run. In order to obviate this problem, an alternative—and more "grid-like"—authentication system has been introduced in the form of GSSKLOG[12]. GSSKLOG was developed by Doug Engert and provides a way to obtain an AFS token by using an implementation of the Generic Security Service (GSS) API; in particular, it uses the GSS-API provided by the Globus Toolkit[13] to obtain an AFS token using an X.509 credential. It comprises two components: a server daemon (`gssklogd`) and a client application (`gssklog`). The server maps the distinguished name (DN) of incoming requests to an AFS user account using a map-file (very similar to the `grid-mapfile` used by Globus [77]), which contains a list of DNs and the corresponding AFS account name. If necessary, more than one AFS account may be listed for each DN, allowing the user to choose which account to use by passing this as an argument to `gssklog`. GSSKLOG has been used successfully in previous work, such as the second *Enabling Grids for E-sciencE* (EGEE-II) project [30].

AFS also provides users with the option to encrypt all communications between client and server, ensuring that data can be transferred securely without fear of it falling into the hands of eavesdroppers. Encryption is controlled by a client-side option that is disabled by default as it places an overhead on AFS operations. There are no server-side encryption options: it is unfortunately not possible to mandate that encryption be turned on before clients are granted access to certain directories.

[12] ftp://achilles.ctd.anl.gov/pub/DEE/gssklog-0.11.tar
[13] http://www.globus.org/toolkit/

Access control under AFS is provided in the form of Access Control Lists (ACLs) which associate AFS users and groups—which are, in effect, very similar to traditional UNIX users and groups—with particular permissions. ACLs are implemented at the directory-level, with all files in a given directory sharing a common ACL (the introduction of ACLs at the file level is on the OpenAFS road map[14]). Users who create a directory are deemed to have ownership of that directory, and can modify the corresponding ACL as they wish; directory owners—although unable to create users or groups—can also permit other users to administer the directory's ACL, and in this sense the authorization mechanism provided by AFS is decentralized [174]. The creation of ACLs can be simplified by assigning users to groups, allowing, for example, all the users from one department or institution to be assigned a common set of permissions on a particular directory. Furthermore, by grouping users based on their job within the project, an authorization model akin to Role-Based Access Control could be developed [173]. The main limitation here is that AFS only supports "supergroups" (i.e. groups of groups) in later releases, and even then this is an option that is disabled by default on many platforms. Without supergroup support, it becomes more complicated to create a role that possesses the privileges of several others. However, one advantage of AFS over traditional implementations of RBAC authorization is that users can be given additional permissions to augment those they receive from their assigned roles [201]; alternatively, negative permissions can be used to create a "restricted" role for a particular user.

In addition to those groups created by users, there are several standard groups defined by AFS itself. `system:administrators` contains those AFS users who possess the conventional administrative attributes of omniscience and omnipotence, while the groups `system:anyuser` and `system:authuser` may be used to refer to all AFS users—including anonymous users—and all authenticated users respectively.

The final benefit of AFS is perhaps the most significant: the availability of a distributed file system providing global access to file-based data using a common nomenclature obviates the need to stage—that is, to transfer through some mechanism such as FTP—data from the "storage" hosts to the "execution" hosts, and vice versa [52]. Managing this form of staging is not a trivial problem: if a job is to be run on a remote resource, all requisite files must be copied and verified prior to execution; on completion of the job, all output must be copied back to storage and, once it has been established that all necessary results have been copied successfully, removed from the execution machines. With

[14] http://www.openafs.org/roadmap.html

AFS there is obviously still the need to copy data, but this copying is performed between the file servers and the cache managers in the local AFS clients as part of the operation of AFS, and so crucially occurs transparently from the point-of-view of a user or application developer. Nevertheless, a knowledgeable user may wish to take the location of AFS file servers and replicas into consideration when deciding where to run a particularly data-heavy job.

When all client systems have access to the same file-system, the process of ensuring that the latest version of critical files is available is greatly simplified. It is also possible, if care is taken, to use this same system to provide users with access to the results of the jobs they have run. Although also potentially a problem with traditional computation clusters, where a user could potentially log-in and alter the contents or location of files during the execution of a job, it is even more important when AFS is used in this manner to remind users that any changes they make to the working directories of running jobs could have significant consequences for the executing applications, as it is easy to forget that, even though these appear as local directories, they are in fact accessible from any AFS client.

2.4.2 Meta-data

The importance of providing meta-data—i.e. data about data, or properties useful in information access or retrieval [117]—that suitably describes the contents of large-scale data stores is now well understood, and it is generally accepted that search functionality limited to simple free-text searching is no-longer sufficient [28, 188].

Much work has been done on meta-data capture and retrieval, although in many cases this concentrates on developing applications to crawl existing resources, such as web pages, in order to automatically or semi-automatically annotate them with appropriate metadata [98]. An alternative approach is to develop a new meta-data service capable of storing the sort of information required by the research projects in which it might be used, and which will support the necessary security infrastructures from inception, as opposed to having them bolted on later. This approach has been adopted successfully by previous grid projects [182, 209].

2.5 Job Scheduling

Scheduling is the process which "...deals with the allocation of resources to tasks over given time periods [while aiming] to optimize one or more objectives" [167]. When

considering distributed computing projects, these resources are likely to be computational clusters and supercomputers; the tasks will be the jobs which users wish to run. Specifically, grid scheduling is a global operation [41], interested in the allocation of jobs to resources; the allocation of jobs to processors within a given resource is left for the resource itself to manage.

The scheduling of grid computing jobs and the management of related resources is a problem that has received a lot of attention [123]. Scheduling applications on computational grids differs from scheduling applications on traditional compute clusters (so-called massively-parallel processor environments) in several significant ways: the pool of resources to which a grid scheduler has access—but over which it has no control—is drawn from multiple administrative domains, and is constantly changing; the resources exhibit very different performance characteristics, and will often be executing other tasks outside the grid environment [27].

2.5.1 Schedulers and Meta-schedulers

Within the grid computing domain, a *scheduler* is a software service running on a particular resource which decides how submitted jobs are scheduled locally. It is often the case that a 'grid' scheduler will sit atop a more traditional batch scheduler (sometimes referred to as a Local Resource Manager or Local Resource Management System), thereby exposing the facilities of the batch scheduler to remote users.

A *meta-scheduler* exists, as the name implies, at a level higher than that of a scheduler; while a scheduler is concerned with the allocation of jobs to computational elements within a single resource, a meta-scheduler is concerned with allocating jobs to distributed resources. For example, given four jobs to schedule, a meta-scheduler may opt to send two to one cluster, one to a second cluster, and one to a third. A meta-scheduler may run either on a centralized resource, or on a user's client machine [17]. There are advantages and disadvantages to both approaches: centralized schedulers generally have a simpler implementation, but present a single point of failure, and have the potential to become a bottleneck in performance [59]. One additional consideration in the case of distributed schedulers concerns the matter of whether instances of the scheduler will co-operate with each other to form an improved schedule.

2.5.2 Scheduler Organization

Broadly speaking, there are three categories into which all scheduling schemes can be split according to the way the scheduler or schedulers involved are organized: centralized,

hierarchical and decentralized. In a *centralized* system, a single controller is responsible for all scheduling decisions; this style of scheduler is typically relatively easy to manage, and ought to have full knowledge of the state of the system it controls—thereby simplifying scheduling decisions—but suffers from the major drawback, at least from the perspective of grid computing, of a severe lack of scalability. Such schedulers are therefore of limited appeal when considering the problem of scheduling jobs on the grid infrastructure as a whole, but must still be understood because of their prevalence in local execution resources which may form part of that grid.

A *hierarchical* arrangement of schedulers has several advantages, addressing the scalability problem and improving the situation with regards to fault-tolerance (although failure of the top-level scheduler could still prove disastrous). A decentralized arrangement also offers benefits in terms of scalability and fault-tolerance, as well as retaining the autonomy of individual resource sites; this comes at the cost of significantly increased managerial complexity, however. A *decentralized* scheduler operating without global control may also have access to as much information as a centralized system, operating instead on that information which is made available by the resources of which it is aware.

In practice, many grid resources have an *overlay* structure [59]; this simply refers to the fact that grids are, by and large, collections of clusters, with each cluster belonging to one organization and, as a result, being reasonably homogenous and static in its construction. This allows the problem of scheduling to be subdivided: perform a coarse allocation of jobs to clusters, then let the individual cluster schedulers—which are often of centralized design—allocate the jobs to particular resources. (It could be argued that this is no different to the state at present within the clusters, where the centralized scheduler allocates jobs to individual machines, and leaves the operating system on these machines to allocate tasks to processors.)

2.5.3 Scheduling Goals

Scheduling algorithms can be described as being either application-centric or resource-centric. An *application-centric* algorithm attempts to optimize the performance of the application (i.e. the performance of the running jobs); a *resource-centric* algorithm attempts to optimize the performance of the system (i.e. to maximize utilization of the constituent resources) [58]. As the goals of an application-centric system will not necessarily conform to those of a resource-centric system, some ‘independent’ schedulers will act to manage negotiations between the two parties (the user and the resource provider) in an attempt to find a solution amenable to both sides [45].

In reality, the situation with regards to an application-centric scheduler is a little more complex than simply trying to maximize performance of one particular application. A scheduler may be operating not solely on behalf of a single user, but rather for a group of users (such as those belonging to a given virtual organization), and as a result will be optimizing the behaviour of all applications that are executed under the auspices of that organization.

The jobs to be scheduled will be either dependent or independent. When a collection of jobs exhibits *dependency*, there exists a relationship between constituent jobs constrained by some ordering, which restricts the order in which the jobs may be run. In this case, the collection of jobs is, in effect, hierarchical; a job may not run until all its predecessors have completed, and any successors it has may not run until it has completed. A collection of *independent* jobs contains no such relationships, and constituent jobs may be run in any order. The management and scheduling of dependent jobs often falls under the remit of workflow tools [219].

2.5.4 Scheduling Information

Schedulers can be classified as either static or dynamic, according to when the scheduling decisions are made. In a *static* system, information about the job and resource state is evaluated at the time of submission, with the allocation to a particular resource being based solely upon this information; in a *dynamic* system, the state of the resources—which will be constantly changing—may be continually updated until the job is ready to run, with the effect that the choice of execution resource may change. Both static and dynamic schedulers are common within grid computing projects. Dynamic schedulers sacrifice simplicity of design in exchange for the ability to more-readily respond to changing load and service levels [59]. A scheduler which reconfigures its algorithm or parameters according to changes in resource state can be described as *adaptive* [41].

A scheduler built to take advantage of the overlay structure can only be successful if the information on which it bases its scheduling decisions is complete and accurate. This will require assimilation of information from the local resource managers (concerning, for example, the number of jobs in the various queues and the level of network activity), data managers (describing the location and availability of data, and any associated licence restrictions) as well as information from the job submission itself (such as a description of the user's access rights). Depending on the environment, some of this functionality may be offered already; for example, Globus provides the Monitoring and Discovery System

(MDS) [50]. Integrating resources which do not already run specific grid middleware will prove more tricky, in part due to the reluctance of many system administrators to install additional software on their clusters.

An important consideration when developing a scheduler for a grid environment is the matter of data location. Unlike in a traditional batch system, where there is no choice of resource and the cost of access to data is therefore constant, in a grid environment the choice of resource can have a profound impact on the speed at which data can be accessed. The quality and speed of the network infrastructure, the contention on the network, and even the geographic location of compute and storage resources, are all significant—and changeable—factors. Although the geographical location of storage resources is likely to be unchanging, data can be moved or copied, thereby changing the set of resources which must be considered. Data placement and scheduling represents an increasingly important challenge, but has thus far received less attention than that of job scheduling.

The information used during the scheduling process can be obtained from various sources. The Berkeley Database Information Index (BDII) is an LDAP database that provides information about grid resources for the consumption of users and services. Its contents is structured according to the Grid Laboratory Uniform Environment (GLUE) schema, which is an information model developed to represent the types of entity commonly manipulated in grid systems, such as resources, services, endpoints and activities [14]. The BDII was developed as part of the LCG project to provide a production-quality replacement for the earlier Grid Information Index Service (GIIS) [71].

Other grid services exist to provide information about resources. For example, GRIMOIRES (Grid Registry with Metadata-Oriented Interface: Robustness, Efficiency, Security – perhaps one of the most contrived acronyms in existence) is a Java service designed to host semantic descriptions of other services. These descriptions may be annotated with metadata relevant to users and service providers, covering aspects such as quality of service and reputation [215]. GRIMOIRES is compatible with the Universal Description, Discovery and Integration (UDDI) OASIS standard.

2.5.5 Optimality of Schedule

The schedule devised by a particular scheduler can be described as either optimal or suboptimal. An *optimal* schedule is the best possible assignment of jobs to resources with regard to some goal (e.g. to minimise the total time from submission to completion of submitted jobs). Producing an optimal schedule within a grid environment is complicated

by the fact that complete information about the state of all resources is rarely available; of course, the scheduling problem itself is NP-complete [206]. For this reason, most schedulers produce a *suboptimal* schedule: a solution that, if not perfect, is sufficiently good. The evaluation of schedules to determine whether they constitute a suboptimal solution is most often based upon a set of *heuristics*, which are generally less restrictive than those based on the assessment of formal computational models (classed as *approximate* schedulers).

2.6 Job Schedulers and Resource Brokers

2.6.1 The gLite Workload Management System

The Workload Management System (WMS) is part of the gLite[15] project, which itself grew out of the EGEE project. It comprises a series of web services which provide support for discovery, allocation and monitoring of heterogeneous resources [136]. The WMS allows users to create various types of job, including jobs which are in fact collections of other jobs (which may—in the case of a parameter sweep or MPI job—work together, incorporate some degree of dependency, or be completely unrelated). Jobs are specified using the Job Description Language (JDL) [159]. As the WMS is designed to handle long-running computation, a proxy-renewal service is provided to ensure that credentials do not expire before a job has a chance to complete.

The WMS allocates jobs to resources following a match-making process, in which the requirements of the job—as presented in the JDL specification—are matched as best as possible to the available resources, which advertise their capabilities in the form specified specified by the GLUE schema [14]. (In its previous incarnation in other projects, the WMS retrieved such information in real-time; in the current implementation, this information is cached in the *information supermarket*, improving both reliability and efficiency [136].) Where more than one resource meets the specification provided by the job, a ranking function is used to order candidate resources in order to determine the 'best' match (i.e. the resource to which the job will actually be submitted); an element of randomness is included in the resource selection to ensure that the most highly-ranked resource is not selected repeatedly for a large number of jobs submitted before the information system can be updated to reflect the increased load placed upon it. A default ranking function is provided by the WMS, but users are able to provide their own function

[15] http://glite.web.cern.ch/glite/

in the JDL specification; thus, candidate resources can be ordered in a manner appropriate to each particular job. For example, one job might rank resources based on their estimated response time, while another might rank them based on their proximity to a particular data-set or the available network bandwidth.

The WMS can operate in two modes: when using *eager* scheduling, jobs are allocated to a resource or resources for execution as soon as possible, at which point they will generally be queued locally on the resource; when using *lazy* scheduling, jobs remain under the management of the WMS until a resource indicates that it is in a position to accept more work, at which point the WMS will allocate the most suitable job from those pending. The former—which is often referred to as *push* mode—is the most commonly-adopted approach in practice.

2.6.2 Condor

The Condor scheduling system was developed as a system to harvest spare processor capacity in networked environments in order to create the sort of high-throughput computing system that would otherwise only be available by means of a supercomputer, while minimizing any adverse effects on the users of the individual machines within the environment. When it was first introduced in 1988, this provided a mechanism to allocate batch jobs to high-performance workstations at times when they would otherwise be under-used, such as overnight [131]. More recently, code from the Condor project was integrated with the Globus project to form Condor-G, a tool which allows users to access resources from multiple regulatory domains [81] (in other words, to access resources within a grid computing environment). Development is ongoing in order to prepare Condor for the challenges presented by the ever-increasing processing requirements of the LHC [31].

2.6.3 Nimrod

Like Condor, Nimrod is a tool designed to maximize usage of networks of loosely-coupled workstations, with particular attention paid to running parameter sweep jobs. Nimrod allows the creation of what is effectively a distributed supercomputer; this differs from what is traditionally considered to be a supercomputer in that it makes use of loosely-coupled machines, with a slow interconnection network. Such a platform is not suitable for all high-performance parallel computing, but can be extremely effective where work can be divided into more-or-less independent units for which the time spent in computation greatly exceeds that spent in initialization [9].

In order to allow users of the Nimrod platform to take advantage of the benefits offered by computational grids, a version of Nimrod which is able to harness features of the Globus Toolkit for dynamic resource discovery and job dispatching was developed. Known as Nimrod/G, it overcomes limitations in the original Nimrod software, such as the use of a static set of resources and the lack of support for various queue managers, which make it unsuitable for use in the grid computing domain [8].

2.6.4 GridWay

GridWay is a framework which aims to improve the performance of jobs in dynamic grid environments by making use of adaptive scheduling and execution. Built upon the Globus Toolkit, GridWay periodically re-evaluates pending jobs and, in the event that the resource to which a job was previously allocated is no longer deemed to be the most suitable, will re-allocate that job to a different resource. Once a job is running, it may be migrated from one resource to another for reasons of fault-tolerance or in an attempt to achieve improved performance; effective migration of an application may require some degree of modification to the application itself [105]. The adoption of a decentralized approach to scheduling can result in reduced overheads when compared to fully- or partially-centralized schedulers such as the EGEE WMS [207].

GridWay has been used as the basis of other projects, such as efforts to improve scheduling in grid environments by taking account of network characteristics, which should reduce the likelihood of a network becoming overloaded [202].

2.6.5 SPHINX

SPHINX is a framework for managing the execution of complex, scientific applications on grid resources. It concentrates in particular on providing fault-tolerance across distributed resources, having the ability to reschedule tasks which were taking an excessively long time to complete, or which were interrupted due to unplanned system downtime [110]. SPHINX can be integrated with systems such as Condor to make use of existing feature sets.

2.6.6 SAGA

SAGA—the Simple API for Grid Applications—was developed as part of a Global Grid Forum (now the Open Grid Forum) standardization effort which aims to bridge the gap between the services offered by grid middleware, and the needs of the applications which have to use it. It is intended to reduce the burden on application developers by

introducing abstractions to mask the differences between the various assorted pieces of grid middleware which exist [89]. Different middleware is catered for by the provision of a plethora of middleware adaptors; adaptors exist for grid middleware such as the Globus Toolkit and Condor-G, as well as for traditional batch managers such as LSF, PBS and Torque, and also for managing plain SSH connections. Experimental adaptors exist to support middleware which has emerged more recently, such as the Amazon Elastic Compute Cloud (EC2). SAGA is written in C++, and also provides bindings for a number of other popular programming languages, including C, Java and Python [142].

Although initially intended for use by application developers, developers of grid middleware have also made use of SAGA when the need arises to interact with components from different grid infrastructures. An example of this can be found in [172].

2.6.7 Local Job Scheduling

The previous sections have discussed the process of global job scheduling, i.e. the allocation of jobs to resources within a system of interconnected resources. Within any single resource, another layer of scheduling exists; the role of these *local* schedulers is to allocate tasks to the constituent nodes of a resource.

Various applications exist to fulfil the local scheduling requirement, each offering an assortment of features to cater for differing requirements. One popular choice is Grid Engine, which was developed by Sun Microsystems and existed in both commercial, and free and open-source variants. Oracle continued to market a commercial distribution after they acquired Sun in 2010. Uncertainty surrounding Oracle's intentions regarding the future of the product led to a proliferation of projects continuing development of the open-source version; among these are Univa Grid Engine[16], Son of Grid Engine[17] and the Open Grid Scheduler[18].

Alternatives include the Portable Batch System (PBS)—which exists in variants such as TORQUE (the Terascale Open-source Resource and QUEue manager), and which can be integrated with cluster schedulers such as Maui and the Moab HPC Suite in order to provide additional features—and commercial offerings such as Platform LSF (Load Sharing Facility) and IBM's LoadLeveler. Another option is SLURM (the Simple Linux Utility for Resource Management), an open-source scheduler used on many of the world's fastest supercomputers [64, 218].

[16] http://www.univa.com/products/grid-engine/
[17] http://arc.liv.ac.uk/SGE/
[18] http://gridscheduler.sourceforge.net/

As this thesis is interested in global, as opposed to local, scheduling, it is sufficient to note that the variety of local schedulers makes integrating global and local schedulers a non-trivial activity. The form which descriptions of jobs take, and the manner in which information about the state of the resource is made available (if such information is actually made available), vary from application to application, which means attempting to integrate with multiple schedulers is a challenge.

2.7 Previous and Related Work

Previous investigations into some of the factors affecting grid scheduling have been carried out. For example, work has been done to integrate the overheads associated with batch resources into the scheduling decision by attempting to place bounds on the length of time a job will spend queued prior to execution [154], as well as on modelling the performance of the application itself [175]. Other research has focussed on reducing the overheads associated with staging data to and from resources, addressing this issue by replicating data dynamically according to its perceived popularity [198]; this work concentrates on explicit data transfer, however, and is not directly applicable when a network or distributed file system is in use, as in these situations the issues of volume replication and data caching will be more critical. Work has also been done on the mechanisms of resource allocation; for example, a multi-dimensional matchmaking model has been developed which tries to use various pieces of static and dynamic information to allocate jobs to resources [111].

There remain facets of resource management that are still not considered. For example, although it is accepted that resource management on the grid is not just an issue of scheduling, but requires consideration of security, licence management and auditing, these aspects are often left to the users. New ways of managing such issues as software and data licensing have been proposed which aim to meet the peculiar requirements of working within the grid or cloud computing domains, but any resource broker must be aware of these in order for them to be truly useful; indeed, the variety of technologies available means that flexibility is ultimately of key importance. Fully integrating such requirements will most likely improve resource management and scheduler performance, although it must be remembered that an overly complex scheduler implementation is unnecessary, not least because any improvements it makes to the scheduling of jobs will be undermined by the increased overheads associated with its operation [149].

Nevertheless, it is proposed that by examining and satisfying certain additional requirements and constraints, it is possible to improve resource utilization, and ultimately the end-user experience. This is the focus of this research.

2.8 Summary

This chapter has introduced, and provided a brief history of, some of the technologies that are prevalent within modern grid and distributed computing. In order to work successfully in the field, it is necessary to be acquainted with a wide selection of technologies, and that accounts for the variety of topics covered in this chapter. In particular, this chapter has sought to provide an overview of work in the field of job scheduling, and also provide a review of the current state-of-the-art in this realm. The following chapters will delve into this area more deeply, and provide a discussion of how some of the limitations mentioned here may be overcome.

Chapter 3
Proposed Requirement Framework

This chapter discusses the various requirements which place constraints upon jobs in a grid environment, that are presently either ignored by the majority of grid scheduling applications, or are not afforded the appropriate degree of importance. It will then introduce a framework that will allow these constraints to be properly considered during the job submission and resource brokering process, before discussing some of the enhancements that will need to be made to current job submission processes in order to support this framework.

3.1 Types of Requirement

3.1.1 A Note on Terminology

The terminology used to describe collections of jobs varies from one environment to another. Hereafter, a *job* is defined to be a unit of work, considered by a user to be a single entity. A job may comprise multiple units known as *tasks*, which may or may not be suitable for independent execution. A job comprising many tasks may be referred to as an *array job*; a *parameter sweep job* is a subclass of array job in which each task performs the same function, but is initialized with differing parameters.

An *allocation* refers to the assignment of tasks to a particular resource. After this assignment is made and the tasks are submitted to the resource, the actual distribution of pending work among the various execution nodes within the resource becomes the problem of the local resource scheduler.

Execution resources offer a number of *slots* in which tasks may be run by users. A slot represents the capacity to run a single task in the case of multi-threaded or multi-process applications, or indeed a whole job in the case where the job is not divisible into tasks.

The number of slots on a given execution node will usually be configured such that it is equal to the number of processor cores in the machine, with the overall number of slots possessed by a resource being equal to the sum total of the number of processor cores in its computational nodes [6]. If a resource has insufficient free slots to run newly-submitted tasks immediately, these tasks will normally be held in a queue until the requisite number of slots becomes available.

3.1.2 Security Requirements

Limited Credential Lifetime

Gaining access to a remote resource will often necessitate presenting some form of credential (a Kerberos token or X.509 proxy credential, for example). These credentials will often have a relatively short lifespan, and may or may not be renewable beyond this period.

> Example 1 – X.509 Proxy Credential
>
> During the course of its execution, a job requires access to a remote resource in order to download input files and upload results. Access to this resource requires a valid proxy credential. The proxy credential is created at the time the job is submitted. The job must complete before expiration of the credential, otherwise the writing of results will fail.

VO Membership

Access to certain resources will often be restricted to members of a particular virtual organization. In many cases, it may be further restricted to those VO members to whom a particular role has been assigned.

> Example 2 – VO Membership
>
> A particular compute cluster which hosts an expensive 3D visualization toolkit is available to anyone who is a member of the VO for one of three different projects, provided they possess the ‘visualization’ role within that VO.

3.1.3 Licence Requirements

There are multiple requirements that must be considered when dealing with the issue of licence management, due in no small part to the numerous and varied forms in which software licences may be issued. The scope of licences may be further restricted when additional conditions are imposed on the granted licence.

Host Licences

A *host licence* is one which is tied to a particular host, but may be used by any process executed on that host. When selecting a resource on which to run the application, only hosts which possess the necessary licence should be considered.

Site Licences

A *site licence* is one which is available to any host which belongs to a particular site, which could be a particular compute cluster, department or academic institution. This provides the flexibility to run the software on a variety of machines, without needing to go to the expense of purchasing a dedicated licence for each machine.

Floating Licences

A *floating licence* works similarly to a site licence, with the exception that the hosts to which the licence may be issued are not confined to a single site. For instance, all sites belonging to members of a particular virtual organization might be able to acquire licences for a particular software product.

Floating licences, which are more flexible than host or site licences, have been identified as a possible solution to the problem of making expensive commercial simulation software available across the large number of distributed resources typical of a grid or cloud computing environment [216], as it would only be necessary to purchase sufficient licences for the estimated number of users, regardless of how many different sites were to offer processing capacity. Such a solution would, of course, require the co-operation of the software vendors in order to be effective.

Personal Licences

A *personal licence* is one which is tied to a particular user, but which may be used to execute a job on any suitable host. Although this type of licence does not directly mandate use of any particular resource, it is important that candidate resources support the application and provide any additional infrastructure necessary to obtain and maintain a software licence.

Example 3 – Types of Licence

A department has three different licensed applications. The first may be run by anyone, but is tied to a particular machine; users must connect to this machine to run their jobs. The second application may be run on any machine in the department, but must check out an appropriate licence before execution begins; there are insufficient licences for every machine to obtain one simultaneously. The third application, like the second, is made available as part of the standard software installation on all departmental machines, but may only be run by particular members of one research group, each of whom possesses a licence tied to their personal credentials.

Combination Licences

The situation is further complicated by the fact that many licences will in fact be combinations of the above types. For example, a licences may permit a particular user to run an application at his or her home institution, but nowhere else (i.e. it would be a combination of a site and personal licence).

Example 4 – Combination Licences

The department in *Example 3* updates their software subscription, with the effect that the third application can no longer be run on any machine in the department. In addition to the personal licence previously required, the application can now only be run on machines which also have a valid host licence.

Licence Expiration

Few licence grants will be made in perpetuity; most licences will have a predetermined expiration time. In certain situations, there may be some considerable time between the licence request being satisfied and the job being executed, in which case careful consideration must be given to the anticipated length of the job in relation to the remaining lifetime of the licence.

Example 5 – Annual Subscription

A particular suite of applications is provided in return for payment of an annual subscription fee. The new licences issued each year by the software manufacturer contain an expiration date one year hence, although one week's grace is factored in to smooth transition from one licensing period to the next.

Licence Duration

Even if a licence has no predetermined expiry time, it may permit access to an application for only a limited period of time. In this case, it is necessary to have some estimate of the likely run-time of a job in order to make a sensible decision about whether to attempt to run the application.

It should be noted that an estimated run-time in excess of the licence duration does not necessarily mean that the application cannot usefully be executed; if suitable provision has been made, perhaps in the form of a checkpoint-and-resume facility, progress can still be made. However, whether or not this applies in any particular case would have to be specified at the time a job is described.

3.1.4 Timing and Usage Considerations

Deadline

Rarely does a user decide to run jobs with no particular purpose in mind. Most jobs submitted for execution will have a particular impetus, and will commonly be governed by some form of deadline. This may be related to the submission deadline for a conference, or the due date of a deliverable related to a research project or commercial contract. In any case, the user would like to run their jobs in such a way as to maximize the chances of them completing prior to the deadline, or at the very least to be informed early in the process if this is unlikely to be possible.

Restricted Access Window

Although it is easiest to assume that a resource will be available for use at all times, this is not always the case. Resources may only accept jobs at certain times of the day or week. Other resources may accept jobs at any time, but it may be prudent to avoid scheduling jobs to run on them at certain times.

> Example 6 – Resource Available at Restricted Times
>
> A departmental resource is available only to staff during weekday office hours (e.g. 08.00–18.00), but is made available to all staff and students at other times.

Example 7 – Resource Unsuitable for Use at Certain Times

A Condor pool installed on student PCs in a university library is available at all times, but jobs will only be run when a particular machine is not in use (or, more accurately, when its load parameters do not exceed some predetermined level). Although jobs may be submitted to the pool at any time, its responsiveness during term time will be significantly less than during holiday periods.

Example 8 – Resource Unavailable for Temporary Period

A resource is due to undergo maintenance, which will include a period of down-time. Jobs should not be scheduled if they would still be running when the maintenance period is to begin.

Presence of Usage Quotas

Use of a particular resource may be limited by means of usage quotas, applied to individual users or groups of users. Jobs will be permitted to run providing the owner has a quota in excess of zero.

Example 9 – Limited Monthly Usage

A university-owned resource is made available for public access. External users are restricted to 150 CPU hours per month, although a five percent 'overdraft' is permitted.

Usage Quotas with Non-Temporal Basis

Usage quotas may be used to restrict jobs on criteria other than execution time. For example, a user may be permitted to submit only a certain number of jobs, regardless of their combined duration.

Example 10 – Limited Number of Jobs

A department maintains a high-performance server for running a few particular types of simulation. The nature of the supported simulations means that these rarely take more than ten minutes to execute. Each user is limited to running forty simulations per week.

3.1.5 Resource Provision

AFS

When a job has need to source files from distributed resources—for example, those made available by means of AFS—during execution, it must obviously be run on an

execution host which supports this. Unlike the need to retrieve files from the Internet via HTTP or FTP, support for AFS is likely to be much less common. 'Support' for AFS, from the perspective of a particular job, necessitates two things: firstly, that the AFS client be installed and operational, and secondly that the particular cell the job needs to access be configured. This latter criterion will likely require some negotiation with the cluster administrators prior to job submission.

Example 11 – Awareness of AFS

Two departmental clusters support AFS, but while both permit access to the local cell which contains the staff home directories, only one cluster permits access to a smaller cell which is used as a repository for certain valuable results. Consequently, a job which requires access to these results must be directed to the appropriate cluster.

3.1.6 Examples of Requirement Combinations

Rarely will the requirements which affect a given job be confined neatly to one of the categories described in the previous sections (see *Section 3.1 – Types of Requirement*). A realistic example of a typical job will usually require the satisfaction of many requirements before it can be successfully executed. Producing a viable allocation of jobs to resources, let alone one which approaches what might be considered optimal, will necessitate the simultaneous examination of multiple requirements. Given that the environment will likely constantly be changing, expeditious decision making is extremely important.

In order to illustrate this, the following tables provide representative examples of jobs and the resources to which they may be assigned.

Let us assume that a job has the requirements illustrated in *Table 3.1*.

Table 3.1 – The requirements governing a particular job.

Tasks	1,000
Licence Requirements	Application X
Personal Licences Available	Application Z
Duration	60 minutes per task
AFS Cell	`nesc.gla.ac.uk`
AFS Volume	`software`
Deadline	6 days

Given the resource state illustrated in *Table 3.2*, it is clear that no single resource can satisfy all the constraints. Resource A (highlighted) can satisfy the requirements for one

Table 3.2 – A simple resource matrix. Only one resource can provide the necessary licences to run the job.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z

fifth of the tasks, and may at some future time be able to satisfy the requirements for the remainder. This would suggest three possible courses of action:

1. Assign all tasks to resource A. Only 200 tasks will be serviced initially, but additional processors may become free over time.
2. Assign 200 tasks to resource A. Re-evaluate situation at later time.
3. Abort scheduling as not all requirements can be satisfied at this time.

Table 3.3 – Two resources can provide the necessary licences to run the job. Furthermore, there are sufficient free processors to commence submission of all tasks in parallel.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z

Table 3.4 – One resource has sufficient host licences to run part of the job. Floating licences may be used by the other resources to run other parts of the job, although the entire job cannot be run in parallel.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z
Floating Licences	$500 \times$ X		

Had the job specification required application Y instead of X, then the constraints could have been satisfied immediately (see *Table 3.3*). Alternatively, with provision of floating licences for application X, use could have been made of those resources without a specific host licence (see *Table 3.4*). In this case, it would seem sensible to utilize resource B or C, in addition to resource A; although all 1,000 tasks cannot be serviced at once, much greater progress can be made by executing up to 700 tasks simultaneously.

Table 3.5 – Two resources can provide the necessary licences and satisfy the AFS requirements to run the job. Furthermore, there are sufficient free processors to commence submission of all tasks in parallel.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z
Floating Licences		500 × X	
AFS Cells	`nesc.gla.ac.uk`	None	`nesc.gla.ac.uk`
AFS Volume Replicas	None	None	`software`

If the resource matrix is extended to permit consideration of AFS requirements, a more complex position is revealed (see *Table 3.5*). Here, it is clear that only two resources, A and C, can satisfy the requirement for AFS, thereby excluding resource B from consideration. Resource C maintains a local replica of the particular volume required by the job; this could have a beneficial effect on performance, particularly if a large quantity of data must be read or written, although the degree of benefit will also depend on the distance between the resource and the file service which maintains the primary copy of the volume.

Table 3.6 – When quotas are imposed on individual resources, these must be taken into consideration when producing possible schedules.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z
Floating Licences		500 × X	
Usage Quota	200 hours per week	500 hours per week	50 hours per day

If usage restrictions, governing either total usage or times of use, are introduced, the situation becomes yet more complicated (see *Table 3.6*). In this example, 200 tasks, each of one hour's duration, can immediately be submitted to resource A, which can satisfy the licence requirements; however, this will exhaust the quota for this resource. 500 tasks can immediately be submitted to resource B, which will exhaust its quota, and will also consume all floating licences during the period of execution. 50 tasks can be submitted per day to resource C, at times other than when resource B is running jobs. In the best case—in which jobs are executed immediately after being assigned to a resource, and all free processors are used—700 tasks will run in parallel across resources A and B in the first hour, with 50 running on resource C one hour later. The remaining 250 tasks will then run at a rate of 50 per day, with the full job completing six days later.

Table 3.7 – Limiting access to resources further complicates the scheduling process.

Resource	A	B	C
Free Processors	200	800	400
Host Licences	X, Y	Y	Z
Floating Licences		$500 \times$ X	
Usage Quota	200 hours per week	500 hours per week	50 hours per day
Availability		18.00 – 08.00 daily	Off-line for 3 days

Any allocation which is produced may also have to accommodate limitations to the periods during which resources can be used (see *Table 3.7*). In this example, the restriction which permits resources A and B to only be used overnight will not have any affect on the overall progress of the job; the outage on resource C, however, will prevent the job completing within its six-day deadline.

When evaluating the suitability of particular resources, it may be beneficial to consider historical information about the state of the resources. For example, in the first example with the resources shown in *Table 3.2*, it is apparent that only one resource can ever satisfy all the requirements. The likelihood of deriving a satisfactory allocation is therefore dependent on the overall capacity of resource A, and the extent to which it is loaded.

3.1.7 Requirement Summary

The requirements that have been introduced can act as use cases, providing a useful indication of where development efforts must concentrate in order to enhance the experience of job submission for end-users. The remainder of this chapter will show how an appreciation of these requirements can be translated into a design for an improved resource broker for grid job submission.

3.2 Translating Requirements to Design

3.2.1 The Job Submission Process

The process of submitting a job to a grid environment begins with a user 'creating' a job. This typically involves putting together a description of the job in a particular format. This is then fed into a tool which understands the description, and is able either to submit it directly to a particular execution resource—which may be a single host or a computation cluster—or acts as a 'broker', selecting the most appropriate resource or resources, but passing the request onto a further tool to perform the actual submission.

3.2.2 The Job Description

In order to submit a unit of work to a resource for execution, it is first necessary to describe it in some way. Various types of description exist, some of which are more commonly encountered than others. For example, Grid Engine—in all its various guises—uses a form of shell script for its job submissions [196], while the pre-Web Services editions of the Globus Toolkit support documents in the Resource Specification Language (RSL) [86].

The Job Submission Description Language (JSDL) [16] is an XML-based language used to describe the requirements of computational jobs (see *Listing 3.1*). It is a specification of the Open Grid Forum (formerly the Global Grid Forum) and, despite the fact that adoption has not been as widespread as would perhaps have been liked, is the closest thing to a standard format available today. It is extensible, with current additions providing support to express requirements relating to the execution of parameter sweeps [61], high-performance computing applications [106] and SPMD applications [176]. Extensions have also been proposed to support meta-broker architectures, of a similar nature to that developed within this thesis, although the particular features provided were not immediately useful [119]. JSDL is the job description format favoured by applications such as GridSAM [127] and the UNICORE grid infrastructure [11].

The gLite Workload Management System (WMS) of the NGS uses the Job Description Language (JDL) as the primary way in which jobs are described [159], but also offers some support for JSDL. JSDL has been chosen as the specification format for the improved resource broker described in this thesis due to its extensibility, along with its status as a standard format.

Listing 3.1 – A simple JSDL document. Name-spaces have been omitted for clarity.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
          <Executable>/bin/echo</Executable>
          <Argument>Hello World!</Argument>
          <Output>echo.out</Output>
          <Error>echo.err</Error>
       </POSIXApplication>
    </Application>
  </JobDescription>
</JobDefinition>
```

3.2.3 Identifying an Application

In order to determine which resources may be used to execute a particular job, it is necessary to unambiguously identify the applications upon which a job relies. In order for information from different resources to be successfully amalgamated, a convention which mandates that applications are referenced in a standard way must be adopted. At present, no such convention exists.

An application can be suitably identified using the combination of a name and version tag. In order to eliminate the possibility of two different but similarly-named applications clashing, the name will be prefixed with a string describing a standard domain name in reverse order, similar to those commonly used to identify packages in languages such as Java [90]. This will be followed by a forward-slash, and the actual name of the application required (for example, `uk.ac.glasgow/BespokeSimulator`).

The version tag will be a freeform string value, permitting application developers maximum flexibility when describing individual versions of their software. Possible version tags would include `1.4.5`, `2.0-RC1` and `1.1_b1194`.

To support interoperability with existing components which may use a variety of dialectal names, it will be necessary to provide some form of map between these and the 'standard' identifiers described above. For example, the information system to which the NGS's WMS refers uses strings such as `NGS-UEE-PYTHON_2_6` to identify particular applications (in this case, Python 2.6) within the uniform execution environment [208].

Other solutions to the problem of identifying an application are also possible, such as using an Object Identifier (OID) [7] or a Digital Object Identifier (DOI®) [162]; regardless of the scheme adopted by a particular system, it is important to be able to translate identifiers from one form to another.

3.2.4 Additional Information Required for Scheduling

From the above examples, it can be seen that reasonable scheduling decisions can often only be made if additional a priori knowledge is provided, beyond that which would ordinarily be required to run a job. Some of this information can already be reasonably supplied as part of a JSDL document, while extensions would be necessary to provide the rest.

The information required over and above that which is necessitated by the simplest of submission descriptions is as follows:

- An estimate of the run-time of a job.
- A statement describing the usefulness of partial completion of a job or task.
- A description of AFS requirements.
- A description of licence requirements.
- An assertion of the suitability for array jobs to be distributed over multiple disparate execution resources.

Obtaining some of this information may require an initial period of analysis. For example, it may be necessary to run a single test simulation in order to get an idea of the time required for execution, and to then use this value as the basis for a run-time estimate.

3.3 Summary

This chapter has developed the general deficiencies identified in the previous chapter into a specific set of requirements that have an affect on the successful execution of jobs within a grid computing environment. It has shown how current job description formats are insufficient for representing these requirements, and identified those areas which will require extension to remedy this situation. The next chapter will propose a design and implementation that addresses some of these shortcomings.

Chapter 4
Design of an Improved Resource Broker

This chapter discusses the design of a system to assist with job scheduling in a manner that takes into consideration the requirement framework outlined in *Chapter 3 – Proposed Requirement Framework*.

4.1 Architectural Overview

For this research, a service-oriented architecture—of the type which is commonly employed in grid computing—has been adopted [79]. A service has been constructed for each proposed requirement which, when passed a JSDL document, updates the candidate hosts set to reflect the set of hosts which can satisfy the particular requirement for which the service has responsibility. Invocation of these services is co-ordinated by a resource broker—effectively a meta-scheduler—with overall responsibility for the successful submission of the job (see *Figure 4.1*). This resource broker is decentralized, in the sense that it will attempt to improve the situation for jobs about which it is aware, and which must co-exist with other jobs under separate management.

The resource broker can be invoked by means of a client application, which will pass it the JSDL document describing the job to be submitted. This document may be hand-crafted, or be produced by another client or service.

It is likely that a more complicated design will be required in the case of those requirements for which the information about whether they can or cannot be satisfied changes quickly. While the existence of support for a particular AFS cell is likely to be fairly constant (cells are usually only added or removed occasionally, and generally only during

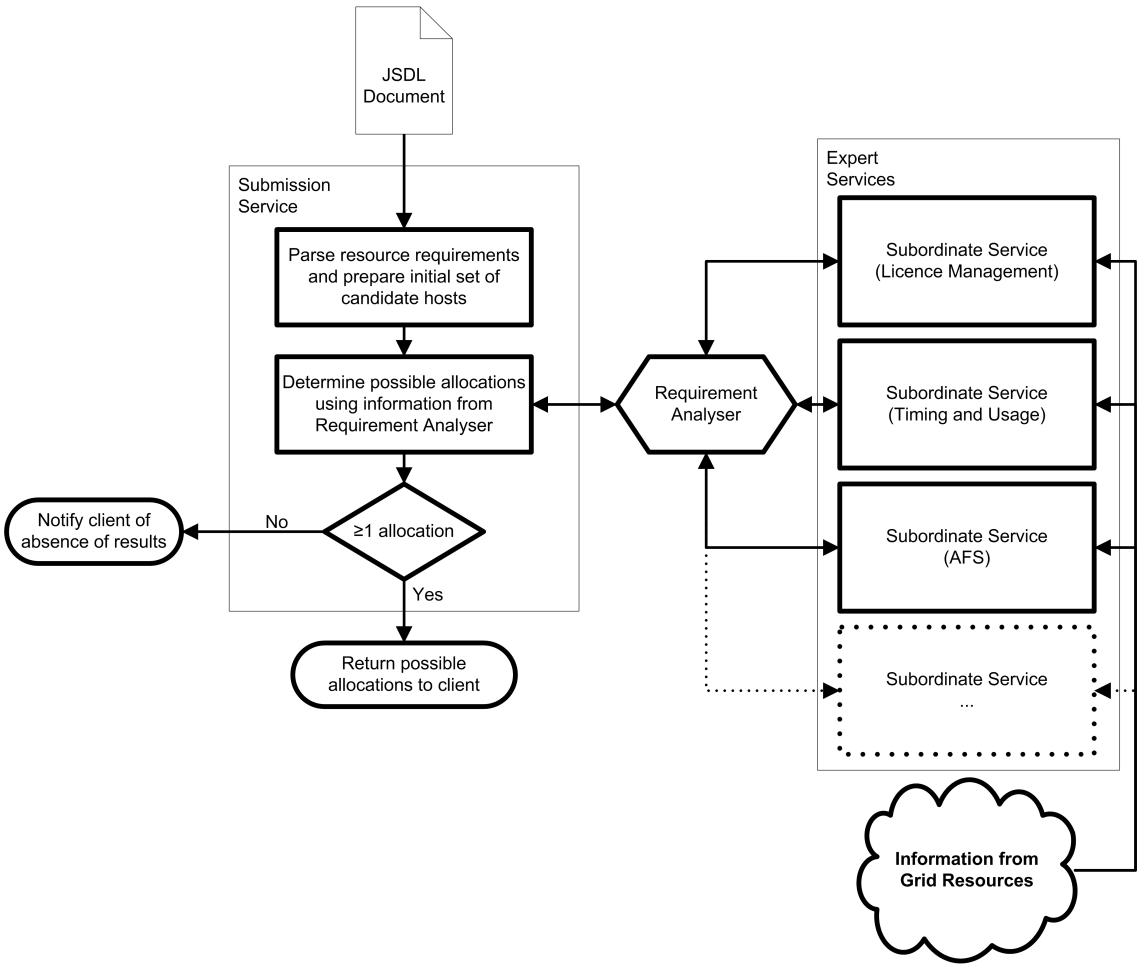

Figure 4.1 – Operation of the requirements analysis service and expert services.

maintenance), the presence of sufficient licences to run a simulation will frequently be in a state of flux. In the latter case, it may be necessary to make some assessment of whether it will still be possible to satisfy the requirement when the job enters the running state, having taken into consideration the total number of execution slots offered by the resource, the number of those slots which are free at the time, and any anticipated time spent queueing.

The use of standards-compliant services serves to provide maximum flexibility with regards to choice of client. In the simplest case, command line applications, which typically entail less development work than graphical applications, may be constructed. Clients featuring rich GUIs or taking the form of web portlets—i.e. browser-based clients providing access via an application server to grid services and resources [152]—could

eventually be constructed for those services which would benefit most from their more user-friendly style of interface; these services will most likely be those with the largest number of users, particularly if a significant number of those users have more limited IT skills.

4.2 Architectural Specification

This section provides a more detailed specification of the design of the requirements analyser within the improved resource broker, as depicted in *Figure 4.1*.

4.2.1 Job Description

Job descriptions are written in JSDL and accord to the version 1.0 specification [16]. Extensions defined in version 1.0 of the HPC Profile Application Extension [106], the Parameter Sweep Job Extension [61], and to a limited extent the SPMD Application Extension [176] are also accepted.

In order to support the collection of the additional information identified in *Section 3.2.4 – Additional Information Required for Scheduling*, the following additions were made to the JSDL schema (see also *Appendix A – JSDL Extension Schema*).

Information about the anticipated run-time of a job can be provided by means of elements such as `<TotalCPUTime>` and `<TotalCPUCount>` (see *Listing 4.1*). The introduction of some estimate of confidence in the anticipated run-time raises the possibility for consideration of the provision of slack time. For example, if CPU time is specified with confidence 1.0 (i.e. the probability of run-time being less than or equal to that specified is one hundred percent), no slack time would be required; if confidence were less than 1.0, rather looser time constraints would need to be imposed (see *Listing 4.2*). Of course, the time required to execute a particular task is dependent upon the speed of the processor on which it is running. The `<IndividualCPUSpeed>` element can be used in combination with the `<IndividualCPUTime>` element in order to specify the anticipated duration given a particular speed of processor. A normalized duration can be calculated as follows:

$$t_{normal} = t_{specified} \times \frac{s_{normal}}{s_{specified}} \tag{4.1}$$

In the above equation, t_{normal} is the normalized duration, $t_{specified}$ is the duration specified in the job description, s_{normal} is the speed of the normalized resource, and $s_{specified}$

the speed of the resource on which the specified duration was measured. Use of the calculated duration assumes that the processors on which jobs are executing are not overloaded; in general, resources will be configured such that the number of slots provided per CPU is equal to the number of physical cores per CPU. If the number of jobs running on a CPU exceeds the number of cores, there will likely be a commensurate drop in performance. Consequently, the calculated duration would be rendered inaccurate, but as such a situation would be considered an error case in a well-configured resource, it is not something that need be handled during the resource allocation process.

Occasionally, situations will be encountered where an estimated duration cannot be supplied with any degree of accuracy. If this is the case, either a rough estimate may be given, or the information may be omitted entirely, both of which may have an adverse effect on the value of any suggested allocation. (If, in reality, a typical task lasts twice as long as its estimated duration, it should come as no surprise if a specified deadline is missed.) There are, happily, many situations—particularly in high-throughput computing—where users wish to submit large numbers of jobs which share behavioural similarities to jobs which have been run previously; any effort made to improve the situation for these jobs will therefore be of benefit, even if examples remain which cannot be handled. In any case, if a user submits a job for which the duration is truly unknown, only a system exhibiting precognition could possibly hope to perform any sort of accurate calculation based on that duration.

Version 1.0 of the JSDL specification discusses scheduling requirements—which includes 'temporal scheduling'—but decrees these to be 'out of scope for JSDL', while stressing the importance of the introduction of a Scheduling Description Language (SDL). This is consistent with the proposals above: information relating to the job—such as its anticipated duration—may correctly be provided in the JSDL document; information relating to the resources—such as their period of operation—should be maintained elsewhere.

Listing 4.1 – Using JSDL to specify computational time requirements. Name-spaces have been omitted for clarity.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
          <Executable>/usr/bin/longsim</Executable>
```

```
          <Argument>-i input.xml</Argument>
          <Output>std.out</Output>
          <Error>std.err</Error>
       </POSIXApplication>
    </Application>
    <Resources>
        <TotalCPUTime>
            <UpperBoundedRange>43200.0</UpperBoundedRange>
        </TotalCPUTime>
    </Resources>
  </JobDescription>
</JobDefinition>
```

Listing 4.2 – Using JSDL to specify computational time requirements with measure of confidence.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
          <Executable>/usr/bin/longsim</Executable>
          <Argument>-i input.xml</Argument>
          <Output>std.out</Output>
          <Error>std.err</Error>
       </POSIXApplication>
    </Application>
    <Resources>
        <TotalCPUTime>
            <Exact>43200.0</Exact>
            <Confidence>0.9</Confidence>
        </TotalCPUTime>
    </Resources>
  </JobDescription>
</JobDefinition>
```

There is currently no natural way in the JSDL specification to encode a requirement for AFS. The best course of action would be to introduce a new element for this purpose (see *Listing 4.3*).

Listing 4.3 – Using JSDL to specify AFS requirements.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
          <Executable>/usr/bin/longsim</Executable>
          <Argument>-i input.xml</Argument>
          <Output>std.out</Output>
          <Error>std.err</Error>
       </POSIXApplication>
    </Application>
    <Resources>
        <!-- Presence of AFS element implies
             need for AFS client -->
        <AFS>
            <!-- Particular cells can be
                 listed as requirements -->
            <Cell>nesc.gla.ac.uk</Cell>
            <!-- May specify whether encryption
                 should be enabled -->
            <Encryption>true</Encryption>
            <!-- Particular volumes can be listed,
                 which would allow scheduling decisions
                 to be based on presence of replicas -->
            <Volume>confidential.data</Volume>
        </AFS>
    </Resources>
  </JobDescription>
</JobDefinition>
```

The present version of the JSDL specification also lacks any natural way to encode the requirement for software licences, although the idea of adding some form of licence description language as an extension to JSDL has been mooted before [129]. These requirements could be included as resource requirements, in much the same way as described above for AFS (see *Listing 4.4*). This particular example includes a request for two instances of a licence; although this might be somewhat unusual, a job can contain a

more-or-less arbitrary set of operations, and so there is no reason in principle why such a situation could not occur.

Listing 4.4 – Using JSDL to specify licence requirements.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
          <Executable>/usr/bin/longsim</Executable>
          <Argument>-i input.xml</Argument>
          <Output>std.out</Output>
          <Error>std.err</Error>
       </POSIXApplication>
    </Application>
    <Resources>
        <Licence>
            <ApplicationName>RestrictedApp</ApplicationName>
            <ApplicationVersion>2.1.1</ApplicationVersion>
            <Count>2</Count>
        </Licence>
    </Resources>
  </JobDescription>
</JobDefinition>
```

If a user has their own personal licence for a restricted application, this information must be conveyed to the resource brokering software. If the personal licence were sufficiently generous as to permit all requested tasks to run at once, then the particular licence requirement could simply be omitted from the job specification, as it would be guaranteed to be satisfied. If, on the other hand, a mixture of personal and other licences were required, it is necessary to know how many personal licences are available during the resource selection process. The existence of personal licences could be included in a JSDL document in much the same way as the specification of software licences (see *Listing 4.5*).

Listing 4.5 – Using JSDL to specify personal licence availability.

```
<JobDefinition>
  <JobDescription>
    <Application>
       <POSIXApplication>
```

```
          <Executable>/usr/bin/longsim</Executable>
          <Argument>-i input.xml</Argument>
          <Output>std.out</Output>
          <Error>std.err</Error>
       </POSIXApplication>
    </Application>
    <Resources>
        <Licence>
            <ApplicationName>RestrictedApp</ApplicationName>
            <ApplicationVersion>2.1.1</ApplicationVersion>
            <Count>2</Count>
        </Licence>
        <PersonalLicence>
            <ApplicationName>RestrictedApp</ApplicationName>
            <ApplicationVersion>2.1.1</ApplicationVersion>
            <Count>5</Count>
        </PersonalLicence>
    </Resources>
  </JobDescription>
</JobDefinition>
```

Version 1.0 of the JSDL specification argues for the introduction of a Job Policy Language (JPL), which would allow users and resource owners to 'apply policies' to their jobs and resources respectively. Unfortunately, it fails to mention what such a policy might be, although it acknowledges that this is a 'very broad subject' and that such policies 'may be applied to any aspect of job execution'. It could be argued that licence management would therefore be best described in terms of job policies (and, hence, would be beyond the scope of JSDL). However, with regard to the licensing of software, the presence of a particular licence is an absolute requirement, as without such a licence the job will not run; it is therefore no more a policy matter than the presence of sufficient memory is a policy matter. Furthermore, if applications licences are being modelled in this manner, it makes sense to model data licences—i.e. restrictions on where certain data-sets may be used—in the same way, lest we end up with multiple disparate components with much the same function. Once again, it should be noted that information pertaining to specific resources—such as the presence of licences for a particular application—should not be stored in the JSDL document.

That being said, it is inevitable that some licensing decisions will necessitate consideration of properties that fall outside the scope of the JSDL specification. For example, mechanisms to secure job submission are considered to be 'out of scope of this specification' [16]. The specification goes on to state that additional, specialized security and policy description languages will likely be needed to express fine-grained delegation of rights. With respect to licence management, it may be necessary to establish a user's home institution or organizational affiliation, in order to determine whether or not he or she ought to be allowed access to a particular application or data-set. This information will not be contained in the JSDL document—rightly, as it does not describe the job, or stipulate a requirement thereof—and so will have to be obtained from other sources. For example, if the submission code were to be contained in a web service, this could be configured to mandate signed communication from the client, and the DN of the signing certificate could be used to determine origin; alternatively, for a submission generated through an appropriately-protected portlet, this information could be retrieved from a SAML assertion provided by Shibboleth.

Candidate Hosts

JSDL provides the means to include a list of 'candidate hosts'—resources which should be considered when determining where the job will be run—within a job description (see *Listing 4.6*). Such a list could be used by a scheduler when evaluating resources for their suitability to run the described job.

When the JSDL document is first received by the consuming service, a candidate host section could be added, populated with the set of resources of which the system is aware. (If the candidate host section were already present in the JSDL document at the time of receipt, this would be taken as an indication of which resources would be considered acceptable to the user, and no additional resources would be added.) Identification of a suitable resource on which to run the job would then become an iterative process: the JSDL would be passed, in turn, to several services, each with a particular remit (one would be responsible for licence management, one for AFS, one for timing, etc.). Each service would remove from the candidate host list any hosts which failed to satisfy the requirements in which it has an interest. Once all services have been consulted, a final scheduling decision would be made based upon some conventional criteria (such as current load on the resource) or in some other manner (such as random selection). Of course, if at any stage it was determined that no resource existed matching all criteria—in other words, if the candidate host set became empty—the submission process would immediately terminate, with appropriate feedback being returned to the user.

Listing 4.6 – JSDL document identifying a set of candidate hosts.

```
<JobDefinition>
  <JobDescription>
    <!-- Application configuration omitted -->
    <Resources>
        <CandidateHosts>
            <HostName>golem.elec.gla.ac.uk</HostName>
            <HostName>lepus.nesc.gla.ac.uk</HostName>
        </CandidateHosts>
    </Resources>
  </JobDescription>
</JobDefinition>
```

Further Considerations

It may prove beneficial to extend the candidate host mechanism to provide some measure of the 'suitability' of a particular resource. This would permit hosts to be ordered in such a way as to improve the final scheduling decision.

When a JSDL document specifies that a job could be regarded as a series of related tasks—as is the case, for example, in a document which contains a parameter sweep specification—it may also be useful for the document to contain some measure of the affinity of the tasks with one another. Such a measurement could be used to limit the extent to which tasks would be submitted to different resources. It should be possible to infer a value for this in certain cases; for example, an MPI job which is defined according to the SPMD extensions would for reasons of performance most likely be best suited to a single resource. (Although there are MPI implementations, such as MPICH-G2 [116], designed specifically to work in grid environments comprising multiple distributed resources, performance would still usually dictate that tasks be kept close to one another if at all possible.) Similar functionality is provided in certain batch schedulers such as the IBM Tivoli Workload Scheduler [109], although in this case it is principally used to mandate that tasks be kept together, rather than encourage their distribution. However, it is not available in the JSDL specifications referred to herein.

It may also be useful to permit the user to mark a job as being suitable for partial execution; this would convey the meaning that, while complete execution of the job is of course desirable, in the event that this is not possible, running the job for a shorter period of time would still permit useful work to be done. For example, a job which features a

checkpoint-and-resume mechanism may be able to make useful progress in quite short periods of time. Alternatively, a long-running simulation which, after producing an initial result simply refines this by improving the degree of accuracy or confidence, may be deemed to be useful provided a certain period of execution is completed.

4.2.2 Resource Broker

The resource broker takes as input a job description as specified above. At present, the resource broker takes the form of a client application which can analyse a user's X.509 proxy credentials—including VOMS attributes, should any be present—to extract additional information upon which resource allocation decisions may be made; such information might include a user's institutional affiliation or virtual organization membership. A useful enhancement might be to re-implement the resource broker as an Apache Axis2 web service, accessible by multiple clients. Such a service ought to require that communication from clients be signed by an X.509 certificate which, in addition to those benefits listed previously, would permit the pool of correspondents to be restricted to those the service administrators wish to trust.

The broker parses any included parameter sweep definition to determine the number of instances of the job—that is, the number of tasks—that need to be run. If a value has been provided for task affinity, this will be used to determine whether individual instances of the job may be run on different resources: an affinity of 0.0 implies that instances may be divided among multiple resources, and affinity of 1.0 implies that instances should be run on the same resource. (A value of 1.0 should probably guarantee that instances will be run on a single resource in order to handle cases where tasks absolutely cannot be divided.) Depending on the particular strategy adopted by the requirements analyser, an affinity value a such that $0.0 < a < 1.0$ may either be rounded to one of these two values, or it may be used as a weighting when the possibility of submission to multiple resources is considered.

The resource broker makes calls to further 'services'—which may or may not be web services in the true definition—to ascertain information about the suitability of candidate resources to receive all or part of the submitted job. Each such service returns information including a list of suitable resources, and possibly some ranking or other indication of the suitability of each resource.

The resource broker returns to the client the results of its deliberations, which will describe one of the following outcomes:

No results (permanent): No allocation could be derived which complies with all specified requirements, and this is due to a failure to satisfy static requirements. Further information may be provided to describe potential allocations in the case that one or more requirements are removed.

No results (temporary): No allocation could be derived which complies with all specified requirements, but this is due to transient limitations (e.g. number of free licences); the situation is therefore likely to change. Further information may be provided to describe potential allocations in the case that one or more requirements are removed.

One result: There exists precisely one allocation which complies with all specified requirements. Details of the resources to which individual tasks should be assigned will be provided.

More than one result: There exists more than one allocation which complies with all specified requirements. Details of each allocation will be provided to the client. Further information ranking the schedules by some order of desirability may also be provided.

4.2.3 Expert Services

The expert services provide information which is useful when considering a particular requirement or category of requirement. These services take the requirement specification, and return a list of resources which are able to satisfy the requirement. The entries on this list may be ranked or weighted in some meaningful way. In certain situations, the list may also include those resources which are able to partially satisfy the requirement.

The operation of the expert services varies according to the sort of information they provide. The simpler services consume file-based data, which may be provided locally or remotely (a report in some structured form published by a web server, for instance); this model is adequate for information which is of a mostly static nature, and also provides a low-cost mechanism for resources to provide information about themselves, without needing to integrate with more complex grid information systems. For example, consider a particular software application that makes use of a legacy licence manager which cannot be modified, and which can only be queried locally. A cron job could be written to query the licence manager once every minute, to parse the output retrieved and to write this information to a file published by a web server installed on the licence server. This

file could be interrogated by an expert service to determine, to within a minute's accuracy, the number of licences available for that particular software package. Given the multitude of licence management solutions which exist [129]—including systems designed specifically to support software licensing in a distributed environment [51, 57], but also many more which are not—this flexibility is very important.

Of course, the ability to extract information from fully-fledged grid information systems is also extremely important; services are therefore provided to interact with the BDII, and which include the LDAP support necessary to do so. There is no reason why multiple sources of information cannot be queried to determine information about a single resource. This would allow major sources of information, such as the BDII, to be supplemented with additional information which may otherwise not be provided. For example, this supplemental information might include details which are not accounted for by the GLUE schema, but which might be particularly pertinent to a certain class of application. Alternatively, it means that the information provided by multiple disparate licence managers can be combined to provide a detailed picture of the availability of licences across a number of resources. The only caveat would be that some order of priority would have to be defined over the information sources, in order to consistently resolve conflicts where various sources disagree about some common parameter.

4.2.4 Analyser

The requirements analyser is the object which encapsulates the knowledge required to evaluate the information provided by the user—in the form of JSDL documents—and the expert services, and to produce an allocation based on this evaluation. The interface of a generic requirements analyser has been defined and standardized, in order that various analysers—each encompassing different strategies—can be swapped in and out as required. In this regard, the component features a modular design.

4.2.5 Client Application

A simple client has been constructed which acts as mediator between users and the resource brokering system. In its present form, it manages the process of passing a JSDL document to the resource broker, and provides functionality to format the results of the brokering operation, presenting these in a format such that the user can choose his or her preferred course of action. A useful improvement would be for this client to provide an interface to assist users in the process of compiling JSDL documents to describe jobs.

4.3 Submission and Enactment

The job description specifies, in general terms, what the job must do. In order to actually execute the job on a particular resource, some component must exist which consumes the JSDL document, and which possesses the necessary knowledge to (a) convert the document into the series of system-specific operations which will permit the job to be run on the selected resource, or (b) convert the document into a different form which can then be further processed by another entity. In the former case, system-specific knowledge concerning the environment to which the job is to be submitted may be required. For example, if a facility exists to reserve resources in order to guarantee the availability of local licences, knowledge of this would be required in order to make correct use of it.

This component may take the form of a submission service—in which case it will execute the job described by the JSDL document—or a brokering service, which will pass the request on to another component for further processing, having possibly performed some translation first.

4.4 Optimization Techniques

It is clearly evident that giving consideration to absolute resource requirements—for example, the presence of a particular AFS cell—will offer an improvement over systems which give no such consideration, and hence may assign jobs to resources which cannot possibly fulfil their requirements. It should also be evident that the ability to divide responsibility for execution of certain array jobs among a series of suitable resources can, in appropriate circumstances, be of benefit.

The simplest solution to the problem of allocating jobs to resources is trivial: process incoming jobs on a first-come first-served basis, allocating them to resources in a round-robin fashion (i.e. the first job would be allocated to the first resource, the second job to the second resource, etc., until all resources have had a job assigned, at which point the process repeats with the next pending job being allocated to the first resource). This is easy to understand, easy to implement, and requires no information about the characteristics of the job or the state of the execution resources.

A simple but effective enhancement to the trivial solution would be to filter candidate hosts based on a series of binary decisions. In other words, all hosts which could not satisfy the resource constraints would be discarded, while the remainder would be considered as equally suitable for allocation.

A more involved allocation process would involve filtering candidate hosts, but then ranking the remaining hosts according to a series of criteria, such as the existence of local replicas of required AFS volumes. This would allow these resources to be ordered in terms of their suitability, with more suitable resources being preferred over less suitable ones.

Once some order of suitability has been determined, this could either be used directly by a simple resource broker, or alternatively the results could be fed into an external broker which already provides advanced allocation and scheduling models. The latter is by far the most beneficial model, as the requirements analysis techniques would influence the final allocation of work to resources, but without needing to duplicate processes which are already mature, and with no need to replace existing infrastructure. For instance, such an approach might make it possible to make use of specialist functionality provided by tools such as Stork [122], a data placement scheduler designed for use with particularly data-intensive jobs.

4.4.1 Proposed Methods

The first requirements analyser to be developed simply filters resources according to whether or not they can satisfy the absolute requirements of a job: do they support the necessary AFS infrastructure? Can a particular licensed application be run there? Does the number of available processors exceed the number of tasks? In the case of resources with no explicit support for array jobs, are measures in place to support these? For example, resources based on Grid Engine can use its native support for array jobs to support parameter sweeps varying an integer parameter according to an arithmetic sequence; supporting more complicated forms of parameter sweep—for example, arbitrary integer sequences or non-integer parameter types—would require additional support beyond that provided by the vanilla batch system (see *Appendix C – Integration with Local Job Managers* for a more detailed discussion of this point). The estimated duration of the job, or its tasks, also needs to be examined. In cases where a job's estimated duration (which is dependent on processor speed) on the resource exceeds its deadline—for example, a job with a maximum credential lifetime of twelve hours, but with a computational requirement of twenty-four hours on that particular resource—the resource will be rejected.

As discussed in *Section 4.2.1 – Job Description*, there will be cases where the duration of the job cannot be estimated. In these circumstances, no attempt to consider deadline or downtime information is made. Where a possible allocation exists, this will still be

returned, and if any of the resources contained therein have periods of downtime scheduled, this will also be noted in order that the user can make the final decision on whether to proceed with submission. If the user has included a deadline specification, a warning to the effect that this was ignored will be presented.

The next step is to adapt this process to consider distributing portions of an array job across multiple resources. Job definitions will either be considered as 'divisible' or not; no particular consideration will be given at this stage to the need to keep tasks close to one another. Once the benefits of dividing such large jobs have been identified, consideration could be given to whether—in a typical scenario with a few large candidate resources—providing some weighting of affinity to the tasks makes any appreciable difference. When distributing portions of a job across various resources, it is of course necessary to pay close attention to the number of instances of a licensed application which can be run at each resource. When personal licences are available, the requirements analyser will use these either to supplement licences already provided at a given resource, or to enable an application to run at a resource which does not otherwise have licences available. Naturally, the application must be installed on the resource—or made available by means of a distributed file system such as AFS—in order for it to be used.

AFS itself is treated in a relatively straight-forward manner. During the initial filtration stage, resources are removed which do not provide access to any requested AFS cells (a step which will also remove any resources which do not support AFS at all), and can be further filtered based on whether or not they are configured to use encryption. The improved resource broker does not attempt to rival the purpose-built data schedulers which have started to come to prominence over the last few years [122]. Instead, it uses a simple model which favours resources where the original AFS cell is hosted, followed by those which maintain a replica of the cell, and finally those which merely provide access to the cell which is hosted elsewhere (albeit with a local cache used to boost performance). If more complex solutions are required, it would be preferable to integrate the tool with a dedicated data scheduler.

This analyser then needs to be extended to consider timing information in more depth. Periods when resources are inoperative or inaccessible will be examined, as will any credential or licence lifetime. In the case of credential lifetime, where resources are provided via the WMS, it is assumed that its proxy-renewal service will be used. Alternatively, where a JSDL document includes a portion according to the MyProxy extension introduced by GridSAM [155], it is assumed that the credential will be renewable through the

referenced MyProxy service at all times until the job's deadline (if specified) is passed. This consideration of timing information can only be useful if some estimate has been provided of the anticipated duration of the job, although it may be possible to rank resources even in the absence of such information.

Considering timing information, which will typically require some prediction of performance, in a distributed context is a complex process [59]. As the analyser exists at a level above that of the particular schedulers running on each resource, it does not have complete information about every job in the system, and so cannot hope to concoct a perfect allocation. Instead, it attempts to do the best it can based on the information it has available. Firstly, it attempts to obtain from the expert services for each resource the total number of slots offered, the number of slots free at the present moment, and information about the typical queueing time. Given this information, it is possible to estimate the time at which a task would begin and end execution on a particular resource. This information can then be further cross-referenced with knowledge of scheduled downtime or job deadlines in order to exclude unsuitable resources. Once the estimates of execution times have been determined, resources are selected in order to minimize the completion time of the job—in jobs comprising multiple tasks, this means minimizing the completion time of the final constituent task—subject to the satisfaction of other requirements. (Deadline and downtime considerations are closely related: internally, a deadline can be thought of as a special case of downtime, in which all resources will become unavailable at the same point in time.)

The analysis process assumes that resources execute tasks in a non–pre-emptive fashion, which is typical of many dedicated computational resources such as those clusters hosted by the University of Glasgow. In other words, once a task has started to run, it will run to completion other than in the case of hardware failure or the meeting of some limit—such as a limit on duration of individual tasks—in which case the task will be terminated. Running tasks will not be suspended or evicted in favour of other, newer tasks deemed to be of higher priority. The degree of support for pre-emption, and the manner in which it is supported, varies considerably from one local resource manager to another. Some systems, such as those based on Grid Engine, have no implicit support for pre-emption [199], while others such as Condor (which was originally designed for cycle-stealing) can make extensive use of pre-emption [49]. Pre-emption can cause significant and unpredictable degradation to the performance of running jobs [60], but any attempt to handle this is beyond the scope of this work.

Where possible, the expert services responsible for providing the licence details offer real-time information—or the best approximation thereof which can be readily achieved—on the current state of licence provision. This may be combined with historical information, if available, to assess the likelihood of a resource being able to satisfy the licence requirement; in this case, the effect of such a change on the analyser would need to be assessed.

The service responsible for maintaining information about the state of AFS can be modified to include details of server and replica locations. The requirements analyser can then take this into consideration when selecting resources; for example, in the case of two resources of otherwise equal suitability, the resource which has a local replica of needed AFS volumes, or which is located closest to such a replica, would be ranked more highly.

4.4.2 Resource Allocation Process

The process by which potential allocations of jobs to resources are derived can be divided into two phases: an initial resource filtering process, in which resources which are manifestly unsuitable are removed (illustrated in *Figure 4.2*), and a later resource selection process, in which the remaining resources are selected based upon additional suitability criteria such as speed (illustrated in *Figure 4.3*).

If at any stage all candidate resources are eliminated, this fact will be reported and all subsequent processing will be aborted.

4.5 Implementation

The improved resource broker broadly speaking comprises three components: a parser, a resource filter, and an analyser.

The parser takes as its input a well-formed JSDL document, and provides a series of methods by which the other components can extract the pertinent information from the job description. Such information might include the applications which are required, the number of tasks to be executed, and a list (if the user has chosen to provide such) of any particular resources on which the jobs may be run.

The resource filter seeks to simplify the analysis step by immediately discounting any resources which are unable to fulfil the request. For example, resources which do not have a required application installed will play no further part in the decision process.

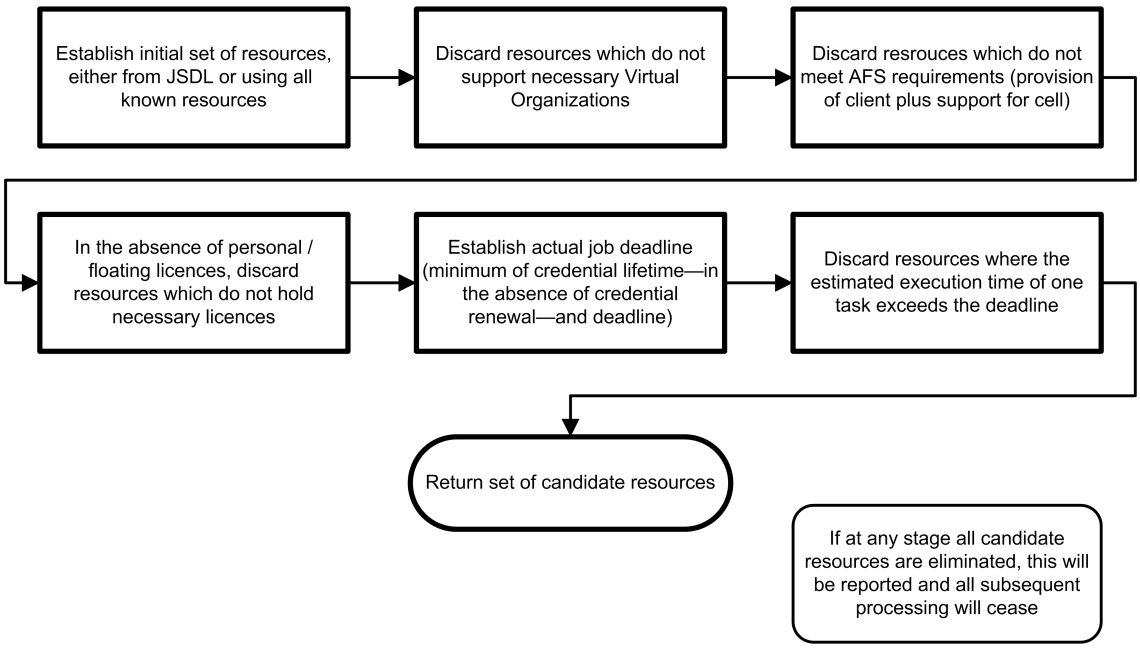

Figure 4.2 – Resource filtering process. Processing will cease if, at any stage in the flowchart, all candidate resources are eliminated.

During this stage, use is made of a series of *application locators*, which report those resources where a given application is available. Presently, three such locators have been implemented: the first maintains a local database of application installations, the second extracts the information from information systems such as the NGS MDS (having first translated the name given in the JSDL to that used by the NGS), while the third combines the results obtained from the first two types. These components have a modular design, so additional locators designed to communicate with other resource advertising systems could be developed and integrated with the existing system with minimal effort.

Assuming that one or more resources survive the filtering step, control is passed to the analyser. The analyser begins by establishing which *licence manager*—the component which provides information about a particular application's licences on a particular resource—is required for each combination of application and resource. It then queries these in turn to build up a matrix which contains information about the distribution of application licences across these resources. Again, these components are modular, and additional implementations may be provided easily to support new or evolving technologies. One or more strategies are then employed to derive a possible distribution of tasks to resources:

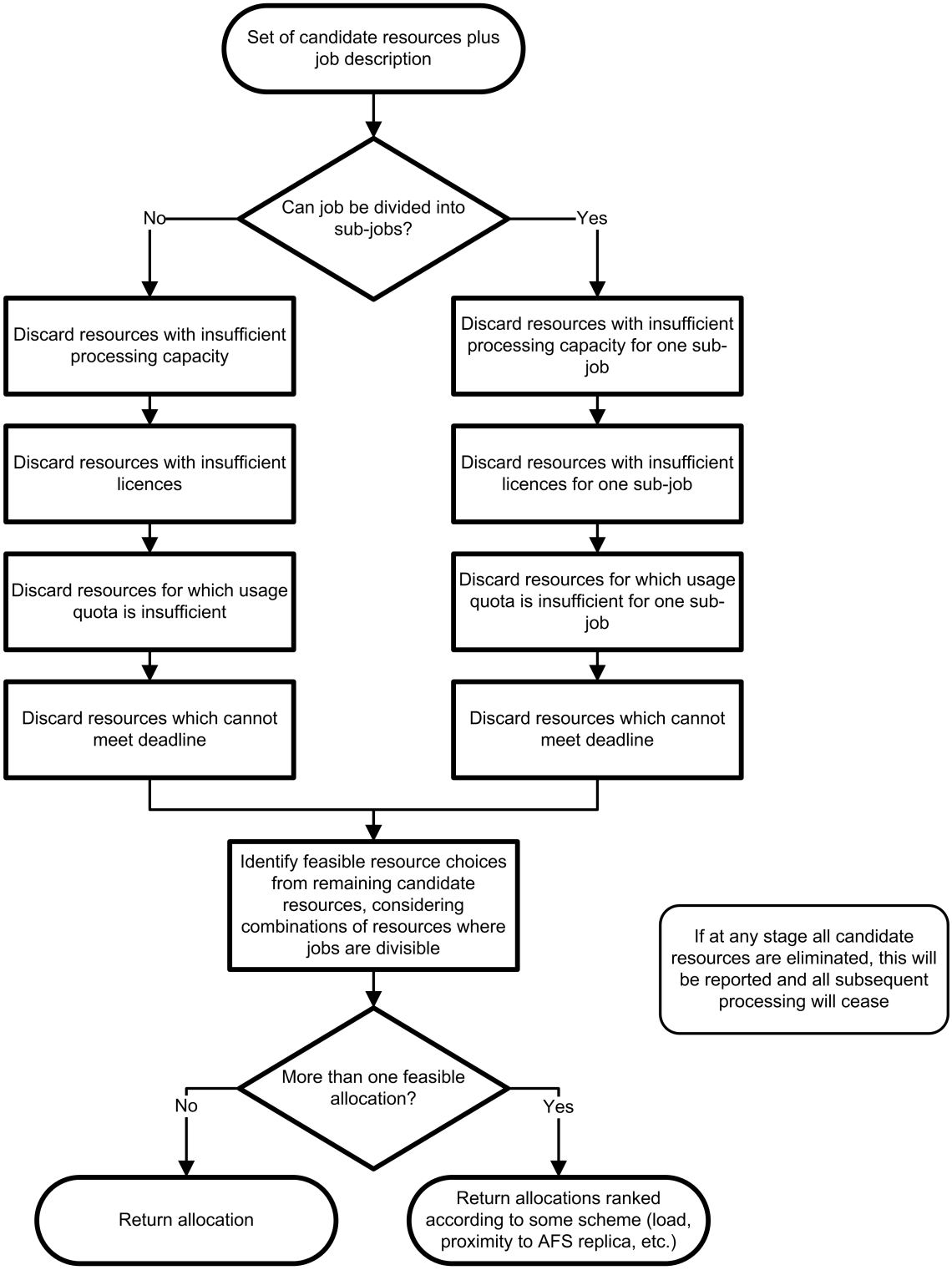

Figure 4.3 – Resource optimization process.

Single Resource: Tasks are allocated to a single resource. Behaviour can be further modified by specifying whether a *best-fit* (jobs are allocated to the resource for which the difference between the number of slots or licences required and number available is the least) or *worst-fit* (jobs are allocated to the resource for which the difference between the number of slots or licences required and the number available is the greatest) allocation is preferred. The validity of a partial allocation—i.e. one where not all tasks are successfully allocated—can also be set.

Multiple Resource: Tasks are allocated to multiple resources. Behaviour can be modified as with the single resource strategy according to whether a best-fit or worst-fit strategy is preferred, and also with regards to the validity of a partial allocation.

Optimization of constraints forms a significant area of computing science research [157, 166], which is largely beyond the scope of this thesis. Many of the requirements considered herein are sufficiently manageable that a heavy-weight solver is not required, and consequently a simple system will suffice.

Some documentation describing the design of the improved resource broker described above is provided in *Appendix B – Program Design*.

4.6 Comparison of the WMS and Improved Resource Broker

Table 4.1 provides a comparison between the gLite WMS, and the Improved Resource Broker described in this thesis. An asterisk (*) in the table denotes a feature that is not currently supported, but which could be added using existing mechanisms.

Notes on entries in *Table 4.1*:

1. The gLite WMS provides a credential renewal service, use of which ensures that a credential will not expire during a long-running job, thereby obviating the need to worry about credential lifetime (other than ensuring that the anticipated duration of the job will not exceed the expiry date of the certificate from which the credential is generated).

Table 4.1 – Comparison between the gLite WMS and the improved resource broker.

	gLite WMS	Improved Resource Broker
Credential Lifetime	N/A (1)	Yes
VO Membership	Yes	Yes
Deadline	No	Yes
Scheduled downtime	No	Yes
Application location	Yes	Yes
Support for AFS	Partial (2)	Yes
Host licences	* (3)	Yes
Site licences	* (3)	Yes
Floating licences	* (4)	Yes
Personal licences	N/A (5)	Yes

2. AFS is included as an information entry in the uniform execution environment. If all cells are not supported by all sites, multiple entries could be used to provide this information.

3. This could be supported relatively easily by including the number of free licences in the information supermarket (in a similar way to that in which the number of free processors is currently recorded).

4. A virtual resource could be created to host floating licences which are not associated with any one physical resource.

5. If it is assumed that a personal licence will always have provision to permit sufficient copies of an application to be run, and that the user will bear responsibility for ensuring this, the licence requirement can simply be ignored.

4.7 Additional Areas for Exploration

There exist possibilities for improving the experience of job submission which have been discussed, but which are not further developed in this thesis.

Hardware Selection

The JSDL specification includes appropriate facilities for specifying the type of hardware on which a job should be run [16]. No attempt is made to make use of this information, although with appropriate support from information providers and

local job managers, it could potentially be useful in certain circumstances, such as when running applications which are optimized for particular processor architectures.

Checkpointing

Specific knowledge of whether or not a job can be check-pointed could be of benefit. The LSF extension schema for JSDL already provides elements to configure checkpointing on LSF resources [15], and this could perhaps be made more generally useful for other resource types. Checkpointing could be useful in various ways, such as permitting a task to do useful work either side of a period of downtime.

4.8 Summary

This chapter has introduced and described the design of a tool to derive improved job allocations in the presence of the requirements outlined earlier in this thesis. It has discussed the resource brokering service, the expert services which support it, as well as, in general terms, the notion of a requirements analyser. It then developed the requirements analyser concept, providing details of the various strategies which are modelled by this component. It has identified the scenarios in which these are likely to be most beneficial, and these scenarios—in conjunction with the use cases described in the previous chapter—will be expanded upon in the next chapter, in which the modes of evaluation are introduced and described.

Chapter 5
Evaluation

This chapter discusses the method to be used for the evaluation of the improved resource broker previously introduced. It begins by discussing the method of simulation to be used, before describing the various tests that will be performed to demonstrate the effectiveness of the requirements analyser. The results from these tests are then presented, analysed and compared to related work.

5.1 Evaluation Design

5.1.1 Measurement versus Simulation

There are two broad classes of method available to evaluate the effectiveness of novel resource allocation strategies. The first is to conduct experiments on live distributed computing resources and to observe and measure the resulting behaviour. This has the advantage of demonstrating that the approach is practical and realistic, as a working system has to be implemented, and developed to the point where it interacts with the live systems alongside which it is being used. There are several limitations to this approach, however, not least that the amount of time which must be invested in the development of a working system is likely to prove prohibitive if multiple potential systems are to be evaluated. The time taken to run realistic jobs—and the cost involved, if access to resources is not provided free-of-charge—is also likely to be prohibitively high [128]. Finally, the state of the resources (for example, their stability and load) cannot be controlled, which can complicate matters when trying to investigate behaviour under particular conditions, and can also make it difficult to repeat experiments using different strategies under the same set of conditions.

Using a simulation to conduct an evaluation of distributed software offers multiple advantages: simulations can often be executed in a small fraction of time required to run the actual applications, and as the system need not be fully implemented—it only needs to work in a way that is representative of the behaviour of a fully-functioning system—the time required, and the costs involved, in conducting evaluations are much reduced.

However, it is essential that if the results of a simulation are to be believable, the simulation itself must be shown to work in a way that is indicative of the behaviour of the real systems it models. For this reason, it is important that care is taken when designing the components of the simulation in order that the outcome be plausible. The evaluation of the improved resource broker was conducted using GridSim.

5.1.2 GridSim

GridSim is a simulation toolkit written in Java using the SIM++ event simulation infrastructure [139]. It can be used to simulate resource allocation in grid computing systems, and in distributed computing systems more generally, whether these lie within one or more administrative domains [37]. GridSim has a number of features of particular use for simulating optimized submission strategies in a grid environment: it allows heterogeneous resources to be modelled, it allows multiple 'users' to submit jobs to a resource simultaneously, and it provides a wealth of statistics that can be analysed to evaluate system performance. GridSim has been used as an evaluation tool in a variety of projects, for example [82, 96, 125, 179].

5.1.3 Modelling Submission Directed by the Improved Resource Broker using GridSim

In order to use GridSim for the evaluation of resource broker improvements, some modifications and enhancements were required. These have been summarised below. A simplified class diagram of this design is shown in *Figure 5.1*.

The `Gridlet` class provided by GridSim models the execution of a job, encapsulating such properties as its length (in terms of millions of instructions), size of input and output, and owner [34]. Objects of the `Gridlet` class also retain a history of the various activities and states through which they pass, allowing a full analysis of the execution of a job to be undertaken once it has completed.

In order to effectively model the various types of job of interest herein, the `LGridlet` class was introduced to extend the `Gridlet` class. The `LGridlet` class has additional

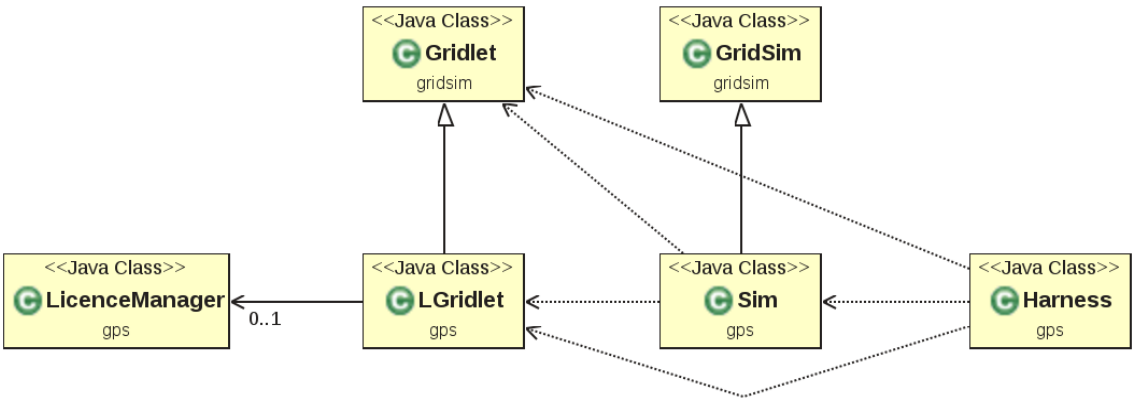

Figure 5.1 – Simplified diagram of simulation class structure. The classes on the top row are provided by GridSim; those on the bottom row were constructed for the evaluation.

properties and methods which allow licence tests to be simulated, deadlines to be checked and resource downtime to be taken into consideration. A new status field provides more detailed information about the final outcome of a particular job; for example, the reason for failure in the case of missed deadlines, resource downtime and insufficient licences is recorded. This permits additional status information to be recorded without the need to modify the underlying status-handling code. Each gridlet also contains a reference to an instance of the `LicenceManager` class, used to determine whether or not sufficient licences exist for a given application on a given resource. These instances invoke an external Web service to determine whether or not the necessary licences are available, simulating the check that a real job would need to make.

The `Sim` class extends the `GridSim` class, responsible for overall management of the simulation. This class creates the various gridlets, loads and allocates timing data, and allocates gridlets to resources according to the chosen policy. This policy may be to use the allocation produced by the improved resource broker, to make a random assignment of gridlets to resources, or to adopt other conceivable mechanisms; this permits a comparison to be made between allocations created by the improved resource broker and those replicating allocations from existing systems.

The final class is the main simulation harness. Its role is to build the model, set the simulation going, and take care of producing the results once the simulation has come to an end. It creates the representations of the various compute resources which will be available to submitted tasks, builds the collections of gridlets, and also determines the

policy which will be used to allocate gridlets to resources. The simulation process is not directly tied to the resource brokering process, although the system state—the load on each resource, the number of free licences at a particular instant, etc.—is common to both. The policy by which gridlets are allocated to resources is configured during initialization of the simulation.

All simulations were executed on a Linux workstation with dual Intel®Xeon®5150 processors (providing a total of four cores at 2.66 GHz) and 16 GB of memory.

5.1.4 Model of Computational Resources

The computational resources modelled within the simulation are based on their real-world counterparts. Although the model is not intended to be a perfect representation of these resources, parameters affecting the size and speed have been chosen to be as accurate as possible based on the information available. The selection of resources used provides a suitable mixture of machines with differing speeds and execution capacities.

The resources chosen are all accessible to the author, with many forming part of the National Grid Service and hence being accessible more generally to academic researchers within the United Kingdom. The resources included within the simulation are listed in *Table 5.1*. It should be noted that, since the resource model was constructed, the NGS node at the University of Leeds has been withdrawn from service.

For the purposes of evaluation, the execution resources are 'pre-loaded' with tasks such that a proportion of the total number of execution slots is occupied when the submission of the tasks under investigation takes place. It is sometimes useful to vary the extent to which resources are pre-loaded with tasks, perhaps to illustrate a particular point that is clearer when the resources are either completely empty or very heavily loaded, and such cases will be noted in the explanatory text accompanying the evaluation.

5.1.5 Limitations of the Simulation

The simulation models the resources identified above, with each resource having the ability to execute a number of simultaneous tasks equal to the number of CPU cores it possesses. Tasks have a duration randomly selected from a set of execution times, and these are adjusted according to the clock speed of the resource on which the job will run, relative to the clock speed of the real system on which the task was profiled. As the task begins executing on a resource, a licence check—if required—is performed and, if this fails, the task is placed immediately in an appropriate failed state. Where deadlines or

Table 5.1 – Resources modelled by the simulation.

Resource	Location	No.	Type	Cores
conan.elec.gla.ac.uk	University of Glasgow	68	12 × 2.66 GHz	816
miffy.elec.gla.ac.uk	University of Glasgow	154	8 × 2.40 GHz	1,232
ngs.leeds.ac.uk	University of Leeds	48	4 × 2.60 GHz	192
		8	8 × 2.60 GHz	64
ngs.ral.ac.uk	Rutherford Appleton	48	4 × 2.40 GHz	192
	Laboratory [130]	8	8 × 2.40 GHz	64
		8	4 × 2.40 GHz	32
ngs.glasgow.ac.uk	University of Glasgow [146]	34	8 × 2.70 GHz	272
ngs.manchester.ac.uk	University of Manchester [147]	64	4 × 2.60 GHz	256
scotgrid.ac.uk	University of Glasgow	108	4 × 2.60 GHz	432
		86	8 × 2.60 GHz	688
		62	16 × 2.60 GHz	992
ngs.lancaster.ac.uk	University of Lancaster [126]	216	8 × 2.26 GHz	1,728
ngs.oxford.ac.uk	University of Oxford [158]	32	8 × 2.60 GHz	256
ngs.wmin.ac.uk	University of Westminster [148]	64	4 × 2.60 GHz	256

downtime are involved, these are checked on completion of the task, and the appropriate state and completion time are assigned. For each task, the resource on which it ran, its submission, start and end times, and the state in which it finished—succeeded or failed and, in the latter case, the reason for the failure—is listed. Random failures associated with factors such as hardware failures are not modelled, as there is no effective way to deal with these prior to the execution of a task, and so they are of little interest here.

The simulation makes no attempt to simulate differences in execution times attributable to CPU architecture, preferring instead to use clock speed as a simple measure of performance. Although it is undoubtedly the case that over the past decade raw clock speed has become less important in determining the performance of a processor, many of the HPC resources available to researchers are of broadly similar age and lineage, and so differences attributable to architecture may be slight. It is also the case that clock speed is a simple numerical metric understandable to users, whereas introducing the notion of processor design complicates matters rather more.

The simulation also does not take into account the quantity or speed of memory present on the execution resources. Although this can be a crucial factor for certain types of job, those profiled and used within the following evaluation require only a moderate

quantity of memory—of the order of 1 GB to 2 GB per process—well within the capabilities of these resources. Where jobs with vastly differing requirements are running alongside one another, it can be helpful to obtain an indication of a job's estimated memory footprint at the time of submission; this can be used to distribute tasks among the nodes of a resource in such a way as to avoid too many high-memory jobs overwhelming a single machine (indeed, the resources within the School of Engineering at the University of Glasgow use just such a system). This is, however, a matter primarily affecting the allocation of tasks within a resource, and so not directly relevant to this work which looks at the allocation of tasks to resources.

Finally, the simulation does not model in detail the differences in network bandwidth within or between the resources. Within the resources, nodes are considered to be connected by a 1 Gbps network (typical of the network speeds of many such resources). A nominal bandwidth is also assigned to the connections between resources. Although this could be of importance to data-intensive jobs, or particularly short jobs where the time spent on staging data to and from the resource may be significant when compared to the execution time of the job itself, it was felt that introducing more detail in this regard would unnecessarily complicate the simulations, while revealing little information of value to the types of jobs being simulated. (It is the case, however, that such detail would be necessary to evaluate fully the functionality concerning AFS, and this is discussed further in *Section 5.3.7 – AFS*.)

5.1.6 Client Application

It should be noted that the client application has been developed principally for evaluation purposes. All communication between client and service makes use of open standards, which aids incorporation of similar functionality in a more comprehensive application.

In order to improve usability, the client application incorporates features designed to simplify some of the attendant tasks associated with running a job on grid resources. For example, the client can handle certificates in a variety of formats—including those generated most commonly by popular Web browsers—and from these it can generate the appropriate grid proxies and VOMS credentials. The only further user participation required is the input of the necessary credential password [209].

One feature which has not been implemented is the provision of a graphical user interface, or GUI, for the client. Although it is still likely the case that a GUI would be

of benefit to those users with more limited IT experience, feedback from the particular research groups working in the domain explored in the evaluation was that this was, at least to begin with, unnecessary [180].

5.2 Modelled Applications

The applications modelled in the evaluation are drawn from the nano-scale semiconductor simulation domain, the area explored by the EPSRC-funded e-Science pilot project *Meeting the Design Challenges of Nano-CMOS Electronics* [19, 186]. To ensure that the model simulates jobs with run-times typical of the applications in which we are interested, timing data for these applications was sampled from some real computational resources.

The number of sample jobs used to generate the run-time data is large. When the simulation is created, each task—that is, each object within the model descended from the `Gridlet` class—is allocated a simulation type. A uniform random selection is then made from the relevant timing data in order to allocate a duration to this task. In this manner, the original distribution of timings is preserved. As long as the timing data can be shown to be typical of the jobs being modelled—which it will be, provided a large enough sample size is used, and obvious anomalies (failures due to unplanned downtime, user error, etc.) excluded—their behaviour within the model will be representative of their behaviour were they to be submitted to a similar execution resource for real.

It is sometimes desirable to create tasks that are already underway at the time the simulation commences. This is of particular use when pre-loading the model with tasks, in order to evaluate behaviour when submitting jobs to busy resources. In the simplest case, a random multiplier may be uniformly selected in the range 0.0 to 1.0 and applied to the task duration, to vary the degree of progress each task made before the simulation began.

5.2.1 Background Information: The Nano-CMOS Application Domain

Over the course of nearly half a century, the prediction made in 1965 by Gordon Moore, co-founder and chairman emeritus of Intel Corporation, that transistor density would double approximately every two years, has been shown to be accurate [143]. Now known popularly as “Moore’s Law”, his prediction describes the relentless strides made in the development of transistor technology.

It is currently common practice when producing circuit and system designs incorporating transistors to assume that all transistors of one type behave in a similar manner; that is, to assume that regardless of the microscopic differences evident in any two physical devices, devices which are macroscopically identical will exhibit behaviour so similar as to be indistinguishable, or, in any case, that differences are so slight as to be considered negligible. As transistors are produced at ever smaller scales, however, the effects of microscopic differences between devices produced to the same design become increasingly more significant; small discrepancies in dimensions, imperfections in the external connections, and even variation in the placement of the constituent atoms themselves, can all have a marked effect on the behaviour of the product. This variability is now recognized by the International Technology Roadmap for Semiconductors (ITRS) [4] as a difficult challenge facing those seeking to manufacture chips in large quantities. With chips now containing billions of transistors, and with transistors with a channel length as little as seven nanometres scheduled to enter production within the next decade, the need for designers to begin taking the variability problem into consideration is becoming ever more urgent [29].

The Device Modelling Group at the University of Glasgow[1] endeavours to provide accurate models detailing the behaviour of semiconductor devices. To do this, process information—which describes the physical, atomic-scale structure of a device, including details of such properties as oxide growth and implantation of dopants—obtained from foundries, and which constitutes intellectual property of enormous value, is used to simulate the behaviour of the device under certain conditions, and in particular to establish the current given particular gate voltages. These simulations are typically extremely time-consuming, necessitating the solving of sets of quantum mechanical and drift diffusion equations which describe the complex interactions of atomic-scale particles within the device. Depending on the use to which the results will be put, it is possible to adjust the complexity of the simulations—for example, by enabling or disabling the simulation of particular effects, or by increasing or decreasing the coarseness of the simulation mesh—with potentially quite significant effects on the amount of computational effort required.

From these results, it is then possible to construct a mathematical model of a device's behaviour by conducting what is, in essence, a rather complicated curve-fitting exercise [48]. The resultant 'compact models' provide a convenient abstraction from which the interaction of multiple devices can be explored without the need to simulate the internal

[1] http://web.eng.gla.ac.uk/groups/devmod

workings of each at the atomic scale. These models can be incorporated within circuit and system simulations, permitting accurate simulation at these higher levels without recourse to huge computational expense.

One of the principal reasons that this domain lends itself well to use in this evaluation is that the applications which are commonly used have differing performance characteristics, permitting scheduling strategies to be evaluated with varied—and, importantly, realistic—jobs. Much work has been done on the development of software to simplify access to these applications [97, 180, 192], as well as in related areas, such as the integration of necessary and appropriate security infrastructure [181], the evaluation and adoption of different file storage mechanisms [25], and the use of various strategies for the management of data and meta-data [97, 186]. The work in this thesis is thus very much grounded in the real world, and aims to satisfy the needs of users in this domain.

Device Simulation

The device simulator, developed in Glasgow, models 3D atomistic devices [18], and is used to simulate ensembles of semiconductor devices. The complexity of the simulations can be adjusted according to the particular effects which are to be investigated, and this has a significant bearing on the computational effort required to complete the simulations.

In general, a simulation of a single device involves very little input and output, but a considerable amount of compute time. For example, input and output will commonly be in the order of 100 kB each, with simulations lasting from half an hour to a couple of days. A Monte Carlo method is used to generate an ensemble of many device simulations. One such ensemble, comprising 100,000 devices, required 15 CPU years of compute time on a 2.4 GHz AMD Opteron system [169].

Circuit Simulation

The circuit simulator is a tool which injects variability-awareness into an underlying SPICE [144] application, thereby permitting those working at the level of circuit simulation to benefit from the lessons in device variability learned at the transistor level. It adds variability by substituting various custom tokens for specific instances of devices drawn from a library of devices modelled according to the variability data. By substituting different instances of a given device and repeating the simulation, a designer can see what the range of likely behaviour will be when the circuit is actually fabricated. This in turn can help the designer develop circuits in such a way as to maximize yield when they are put into production.

The input for a single SPICE simulation is typically small, in the order of hundreds of kilobytes. However, the simulation parameters are hugely customizable, and this affects the quantity of output generated; as a result, output from a single simulation can vary from hundreds of kilobytes to hundreds of gigabytes. It would be very difficult for a non-specialist tool—for example, a generic job submission application—to determine in advance how large the output for a given SPICE job would be. A circuit is stored as a network of components, and the time required for simulation will depend, among other things, on the number of nodes in the circuit. Depending on what it is the designer wishes to evaluate, a single SPICE simulation can take anything from a fraction of a second to many hours to run.

5.2.2 Example Jobs

From the various types of work conducted within the selected application domain, two particular examples were selected for use during the evaluation of the improved resource broker. These examples were selected from jobs running on local execution resources for which the audit logs were available. From these logs, timing data could be extracted which revealed the amount of time required to execute each constituent task. These sample durations were then used during the evaluation to ensure that the behaviour of the simulation was realistic, as described previously in *Section 5.2 – Modelled Applications*.

Type 1: Short Simulation

This example is based upon a large number of relatively simple atomistic device simulations, the purpose of which were to investigate the effects of several sources of variability in a 25 nm PMOS device. As well as looking at different sources of variability, the simulations were repeated with varying external conditions, such as different drain biases.

The completion of any one simulation instance requires various differential equations to be solved. The resolution of the simulation—i.e. the level of granularity at which the user is examining the device—and the type of effects to be investigated have bearing on the number and complexity of equations that must be solved. In this particular example, the simulations were of such limited complexity that the mean duration was a mere thirty-six minutes; by comparison, more complex simulations of a similar nature can take many days to run. The resulting distribution is described in *Table 5.2* and illustrated in *Figure 5.2*. It is apparent from the graph that, while the durations of the majority of tasks form a bell curve, the nature of the work is such that the underlying equations will

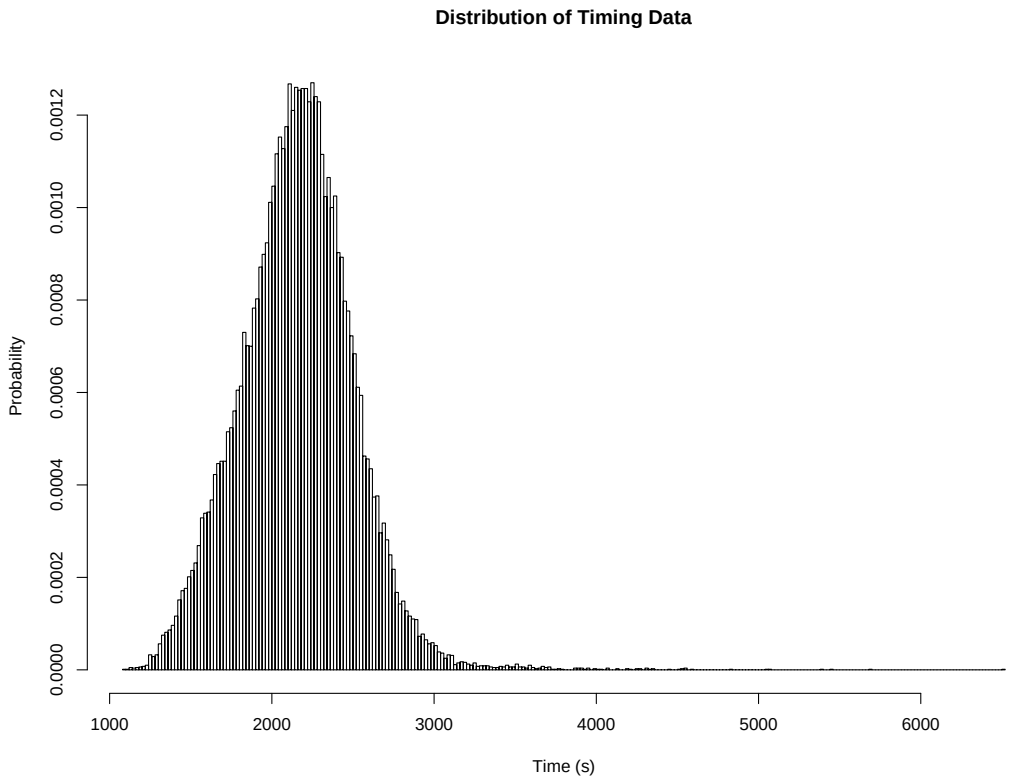

Figure 5.2 – Distribution of run-times of type 1 (short) job.

sometimes take a significantly longer time to converge than usual (or, indeed, will fail to converge entirely). This is the cause of the long tail which is evident in the graph, with some durations lying beyond the 5,000 second mark, well over double the mean duration. Such outliers make it very difficult to guarantee performance in certain situations, a fact which will be observed later.

Table 5.2 – Distribution of run-times of type 1 (short) job.

Samples	40,000
Mean Duration	2,160
Standard Deviation	347

Type 2: Long Simulation

This example is based upon work carried out as part of the MODERN project[2], and was one of many simulations performed to investigate the effects of statistical variability

[2]http://www.eniac-modern.org

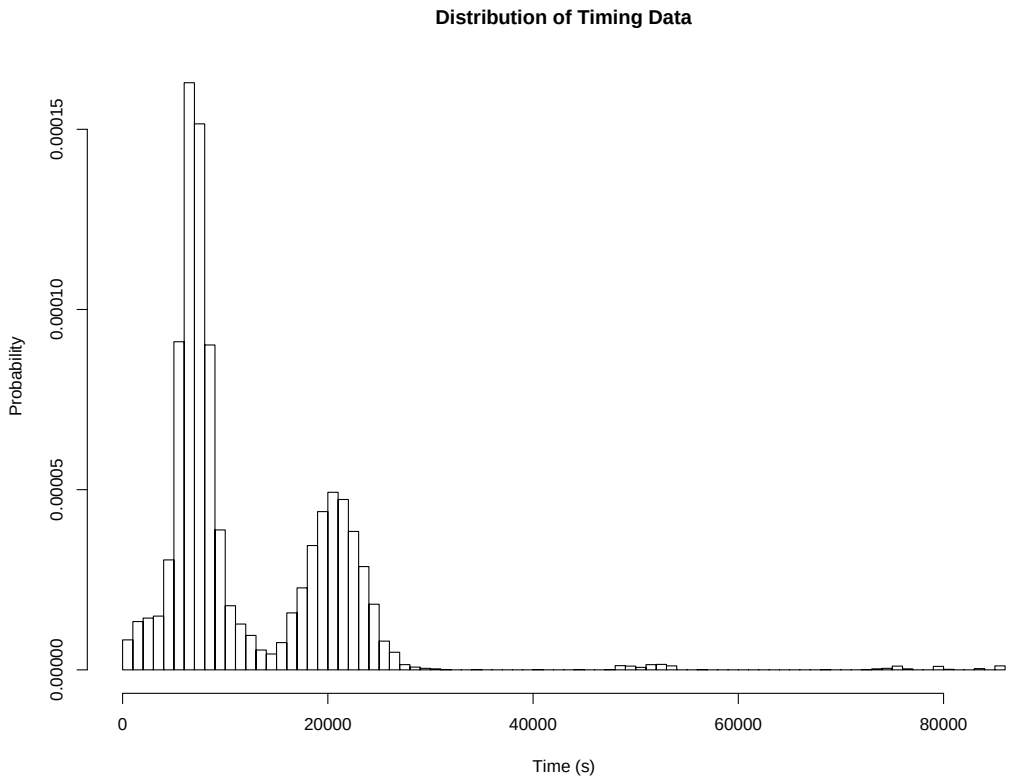

Figure 5.3 – Distribution of run-times of type 2 (long) job.

on a non-volatile flash memory cell (variability results drawn in part from those jobs profiled can be found in publications such as [171]). This is a more complicated atomistic simulation than the previous example, and accordingly the mean duration is longer (see *Table 5.3*); from the graph (see *Figure 5.3*) it is also evident that, due to the intricacies of the simulation, there is a distinct double-peak in the distribution of task durations. Consequently, estimating the duration of a task is more complicated than when the distribution has a more common shape (for example, one which approximates a bell curve, as in the first example). Furthermore, the distribution also features a long tail with several outliers, indicating tasks where the mathematical solvers took a much longer time than usual to reach a solution.

Table 5.3 – Distribution of run-times of type 2 (long) job.

Samples	15,000
Mean Duration	12,108
Standard Deviation	8,897

5.3 Evaluation Results

In order to investigate the change of task behaviour when the improved resource broker is introduced—and to demonstrate an improvement in this behaviour—a series of tests was planned and carried out. A list of tests, along with the intended purpose of each, is provided, followed by a summary of the results. Further numerical data may be found in *Appendix E – Evaluation Results*.

5.3.1 List of Experiments

A series of experiments was devised and performed, in order to test various aspects of the resource brokerage strategy. Each experiment was repeated twenty times in order to ensure reproducibility of results, and to reduce or eliminate discrepancies introduced by the random sampling of execution times. These experiments were as follows:

WMS-style submission using type 1 and 2 jobs

This experiment demonstrates the behaviour of the system as it exists at present, and gives a basis for comparison later. 1,000 tasks were used. See *Table E.1* and *Table E.2* for summarized numerical data.

IRB submission using type 1 and 2 jobs

This experiment demonstrates the behaviour of the modified system, particularly as regards varying affinity. 1,000 tasks were used. See *Table E.3* and *Table E.4*.

WMS-style submission of licensed tasks using type 1 and 2 jobs and site licences

This experiment demonstrates the behaviour of the system at present when used with jobs which require some degree of licence management. 1,000 tasks were used, with resources either holding site licences or no licence at all. See *Table E.5* and *Table E.6*.

WMS-style submission of licensed tasks using type 1 jobs

This experiment demonstrates the behaviour of the system at present when used with jobs which require some degree of licence management. 1,000 tasks were used, with resources not necessarily holding enough licences for all execution slots to acquire one simultaneously. See *Table E.7*.

IRB submission of licensed tasks using type 1 and 2 jobs

This experiment demonstrates the behaviour of the modified system when used with jobs which require some degree of licence management. 1,000 tasks were

used, with resources either holding site licences or no licence at all. See *Table E.8* and *Table E.9*.

IRB submission of licensed tasks using type 1 jobs

This experiment demonstrates the behaviour of the modified system when used with jobs which require some degree of licence management. 1,000 tasks were used, with resources not necessarily holding enough licences for all execution slots to acquire one simultaneously. See *Table E.10*.

WMS-style submission of tasks using type 1 and 2 jobs with deadlines

This experiment demonstrates the behaviour of the system at present when used with jobs with some deadline specified. 1,000 tasks were used. See *Table E.11* and *Table E.12*.

IRB submission of tasks using type 1 and 2 jobs with deadlines

This experiment demonstrates the behaviour of the improved system when used with jobs with some deadline specified. The increase in percentage of jobs completed prior to the deadline is of particular interest. 1,000 tasks were used. See *Table E.13* and *Table E.14*.

WMS-style submission of tasks using type 1 and 2 jobs with downtime

This experiment demonstrates the behaviour of the system at present when used with jobs with planned downtime declared. 1,000 tasks were used. See *Table E.15* and *Table E.16*.

IRB submission of tasks using type 1 and 2 jobs with downtime

This experiment demonstrates the behaviour of the improved system when used with jobs with planned downtime declared. The increase in percentage of jobs completed will be of particular interest. 1,000 tasks were used. See *Table E.17* and *Table E.19*.

WMS-style submission of tasks using type 2 jobs with various requirements

This experiment demonstrates the behaviour of the system at present when used with jobs in scenarios with various requirements and conditions. Both 1,000 and 2,000 tasks were used. See *Table E.20* and *Table E.22*.

IRB submission of tasks using type 2 jobs with various requirements

This experiment demonstrates the behaviour of the improved system when used with jobs in scenarios with various requirements and conditions. The increase in

percentage of jobs completed will be of particular interest. Both 1,000 and 2,000 tasks were used. See *Table E.21* and *Table E.23*.

WMS-style submission of tasks using type 1 jobs with various requirements

This experiment demonstrates the behaviour of the system at present when used with jobs in scenarios with various requirements and conditions. This test was based on the previous 1,000 task test, with the allocation and load from the previous test forming the baseline load in this test. 1,000 tasks were used. See *Table E.24*.

IRB submission of tasks using type 1 jobs with various requirements

This experiment demonstrates the behaviour of the improved system when used with jobs in scenarios with various requirements and conditions. This test was based on the previous 1,000 task test, with the allocation and load from the previous test forming the baseline load in this test. 1,000 tasks were used. See *Table E.25*.

Jobs mostly comprising 1,000 tasks were used because this figure is a common choice for simulation ensembles produced with the applications modelled in this evaluation. In an examination of 2,836 such jobs, 1,426 (50.3%) were single simulations containing a lone task, many of which were run for the purposes of software development or simulation calibration. 705 (24.9%) were 1,000-task jobs similar to those used in this evaluation, while the remainder ranged in size from two tasks to ten thousand tasks.

5.3.2 Varying Affinity

Current Operation

An evaluation of the benefits and potential improvements offered by the improved resource broker over existing systems must begin by demonstrating the behaviour of these existing systems.

In the simplest case, when submitting a job using the WMS the execution resource is selected more or less at random [145]. Some basic information is taken into consideration; for example, the VOs to which the submitting user belongs are examined, and only sites which support these VOs are used. A site which supports a particular application can be chosen if that application has been registered appropriately. The availability of licences can be considered on an ad hoc basis, although there is no standard framework for managing such things, and as such this is largely left up to system and VO administrators to deal with. More complex requirements such as deadlines and projected downtime for the execution resources are also not taken into account.

The 'random' selection of a site can be influenced at submission time by the user including appropriate parameters in the job description. Such parameters tend to work at a fairly low level; for example, it is possible to mandate that a job be submitted to the least heavily-loaded resource. This will not necessarily be the best site for the job; it may, for instance, be better to choose a busier, but higher-performing, resource. This also requires the user to have some understanding of the mechanics of the resource allocation process. Significantly, the matter of spreading a job divisible into smaller units of work across multiple resources is left entirely to the user.

The result of submitting 1,000 independent type 1 tasks in this manner is presented in *Table E.1*, which as with all similar tables may be found in *Appendix E – Evaluation Results*. For this test, no licence, deadline or other restrictions were imposed. As with subsequent tests, slots on resources were filled with similar tasks, each at a random point in its execution, in order to simulate typical load. The results of submitting 1,000 independent type 2 tasks is presented in *Table E.2*.

Improved Resource Broker

When the improved resource broker is used to select an appropriate execution resource or set of resources, in this case it selects those that will likely execute the job in the shortest possible time. (The situation becomes more complicated in later tests where faster resources may be discarded if it is felt there is too great a risk that downtime will interrupt running tasks, or if sufficient licences or other resources are unavailable.) When

a submission is made with an affinity of 0.0—in other words, when there is no requirement to keep component tasks on the same resource—tasks will be allocated to resources according to an estimate of the overall task duration, in an attempt that this value be minimized. In this particular case, the suggested course of action is to submit 816 tasks to `conan.elec.gla.ac.uk` and the remainder to `ngs.glasgow.ac.uk` (*Table E.3*).

The results show that the overall success rate is maintained (*Table 5.4*), and that each task executes more quickly on average, as would be expected since the tasks are always being distributed amongst the fastest nodes. (As all resources were loaded to an equivalent degree prior to selection, the penalty in terms of queue times is broadly similar, depending only on the speed of the resource.) The results for the type 1 (short) test jobs show a decrease in mean duration of 1,823 seconds (over half an hour), which equates to an improvement of 35.3%; these results are illustrated in *Figure 5.4*, *Figure 5.5* and *Figure 5.6*. Regardless of the execution resource selected, all tasks completed successfully.

Table 5.4 – Job outcome: affinity (type 1 tasks).

	WMS	**IRB**
Completed	100%	100%

In the example drawing on type 2 (long) tasks, a similar allocation of jobs to resources is proposed; resultant data is summarized in *Table E.4*. As with the short example, all jobs completed successfully (*Table 5.5*). Task duration and completion is illustrated by *Figure 5.7*, *Figure 5.8* and *Figure 5.9*. Under the improved resource broker, mean task duration is seen to be reduced from 23,189 seconds to 14,905 seconds (a reduction of 35.7%, very similar to that seen in the first example).

Table 5.5 – Job outcome: affinity (type 2 tasks).

	WMS	**IRB**
Completed	100%	100%

No attempt has been made to make use of affinities other than 0.0 or 1.0, although the affinity setting has been defined as a floating-point type with the intention that other values could be used. It might be interesting in the future to investigate whether a setting of, for example, 0.5 could be used to indicate that tasks may be divided amongst resources, but that the total number of resources used should be kept as small as possible. In an ideal world this degree of control is perhaps unnecessary, as a user would typically either

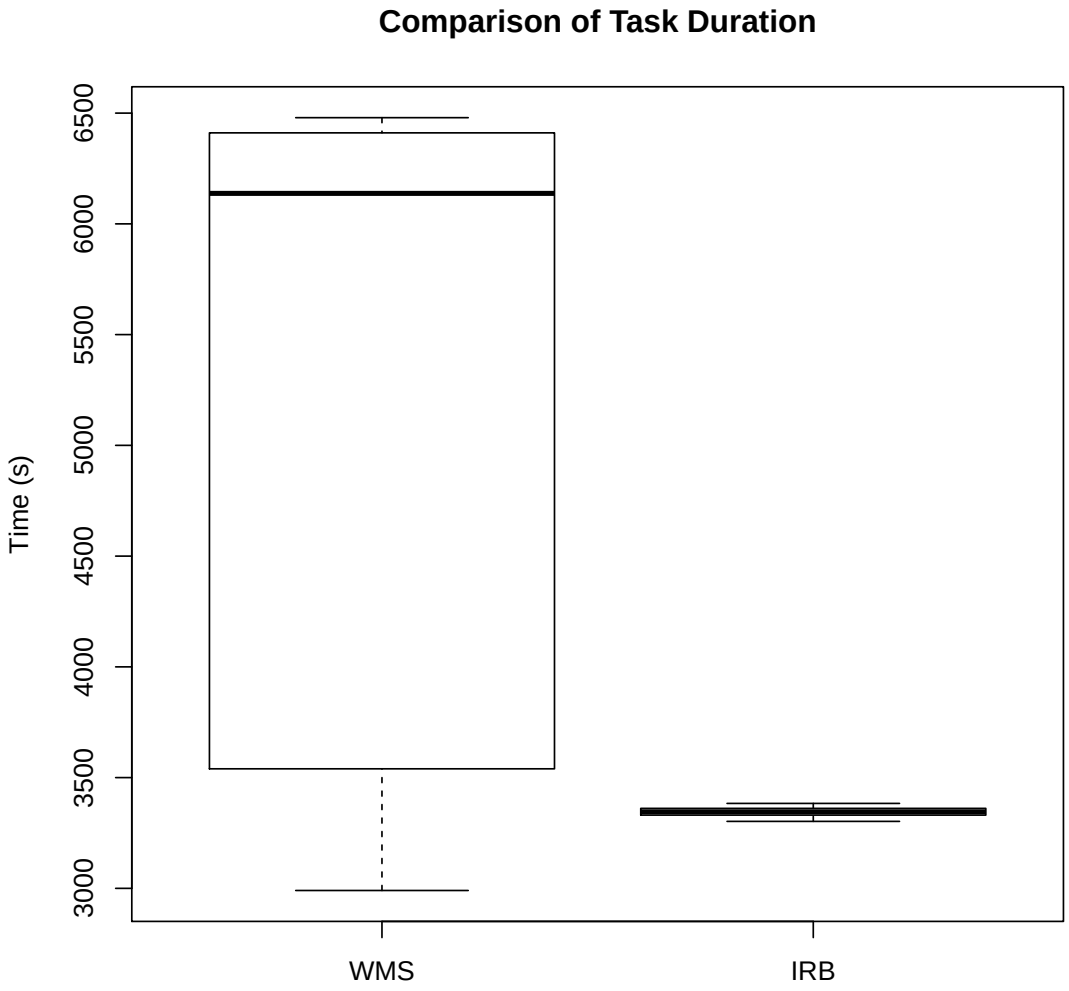

Figure 5.4 – Comparison of duration of jobs (type 1 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

mandate that tasks be kept together for performance reasons (e.g. if they were part of an MPI job) or specify that they could be distributed amongst all available resources. However, there are situations where the ability to take advantage of more than one resource would be beneficial, while at the same time restricting the overall number of different resources used. For example, it is often easier to debug a new application if the results are constrained to a limited number of locations, but it may be unfeasible to use just a single

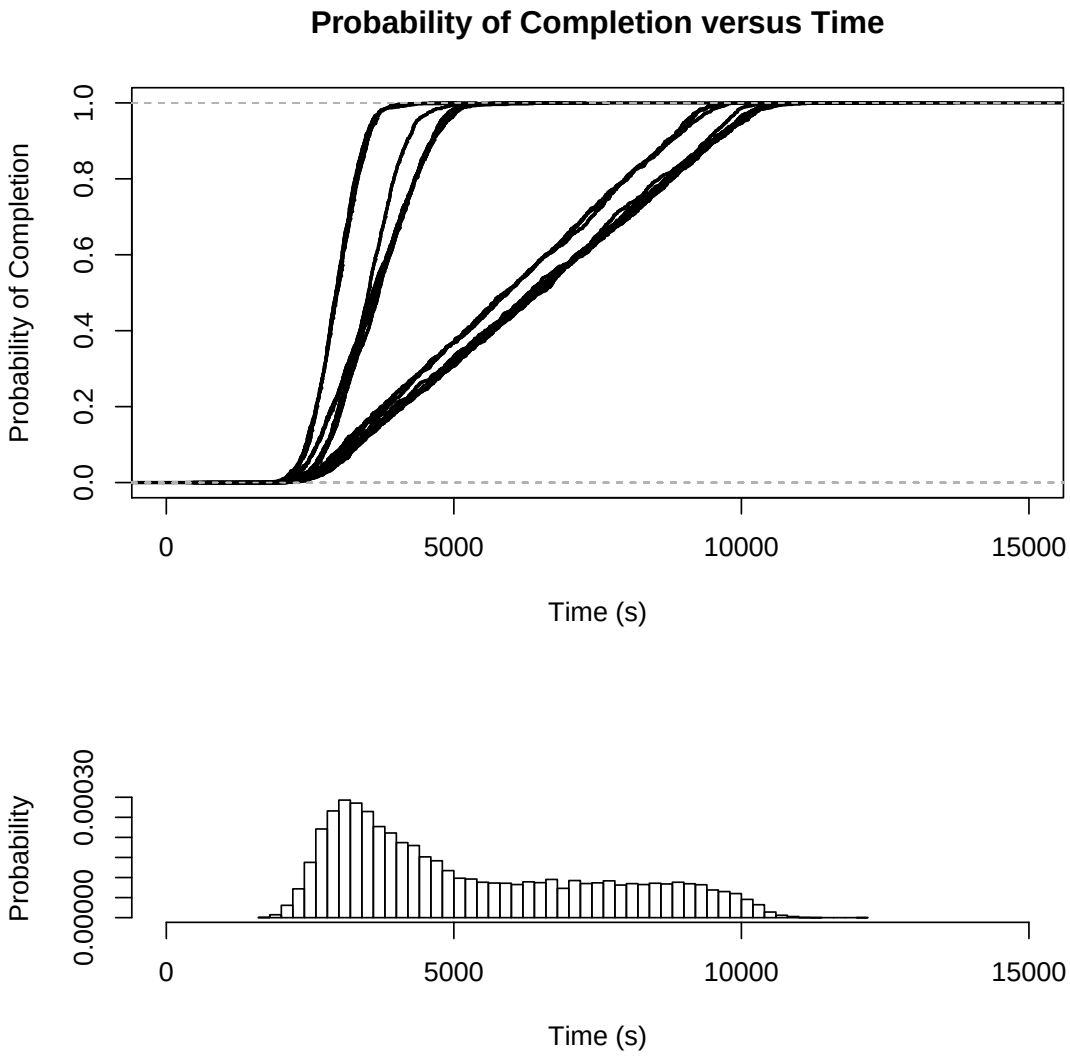

Figure 5.5 – Execution profile of jobs (WMS, type 1 tasks). The same data is illustrated both as a cumulative distribution function (upper graph) and as a histogram (lower graph). In this particular example, it is clear that no tasks finish until about 2,000 seconds have elapsed, and that all tasks finish shortly after 11,000 seconds have elapsed. The distinct lines in the upper graph describe the progress of tasks on resources of differing speeds; faster resources execute tasks more quickly, which results in a steeper gradient.

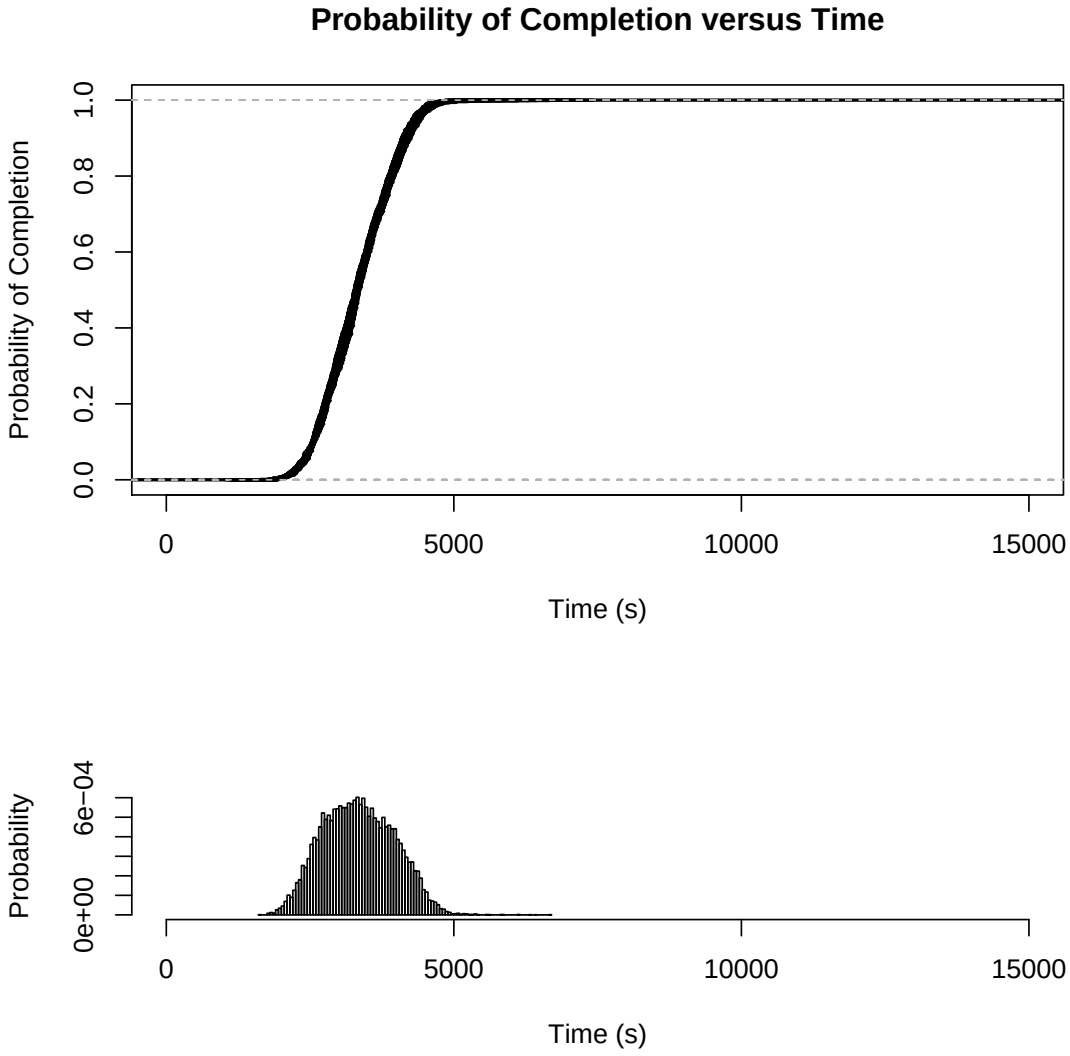

Figure 5.6 – Execution profile of jobs (IRB, type 1 tasks). In the absence of factors such as licensing concerns, the improved resource broker targets the fastest resources, dividing suitable tasks among more than one resource if possible. Consequently, it is evident that behaviour is more consistent than in the previous example.

resource if the resources are particularly busy, or if a problem to be tested only presents itself, or is exacerbated, when multiple resources are used. Alternatively, while it has been noted that the constituent processes of an MPI job should ideally be located closely together, MPI implementations such as MPICH-G2 exist which have been specifically designed for use in grid computing, and where such an implementation is used it may be preferable to make use of a small number of different resources rather than being forced to wait for sufficient execution slots to become available on one particular resource. In this case, a high affinity (but less than 1.0) should indicate that distribution is permissable, but that selection of a single resource would be more desirable.

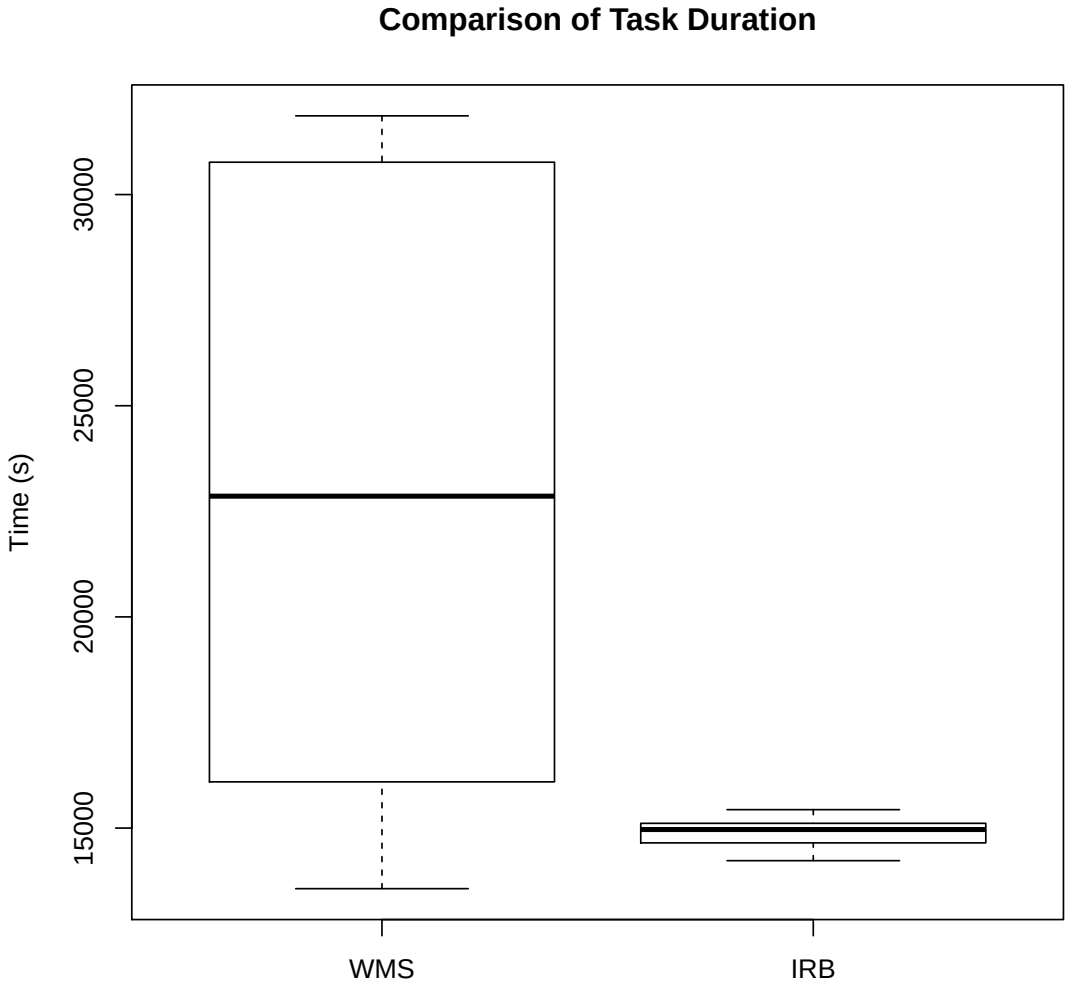

Figure 5.7 – Comparison of duration of jobs (type 2 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

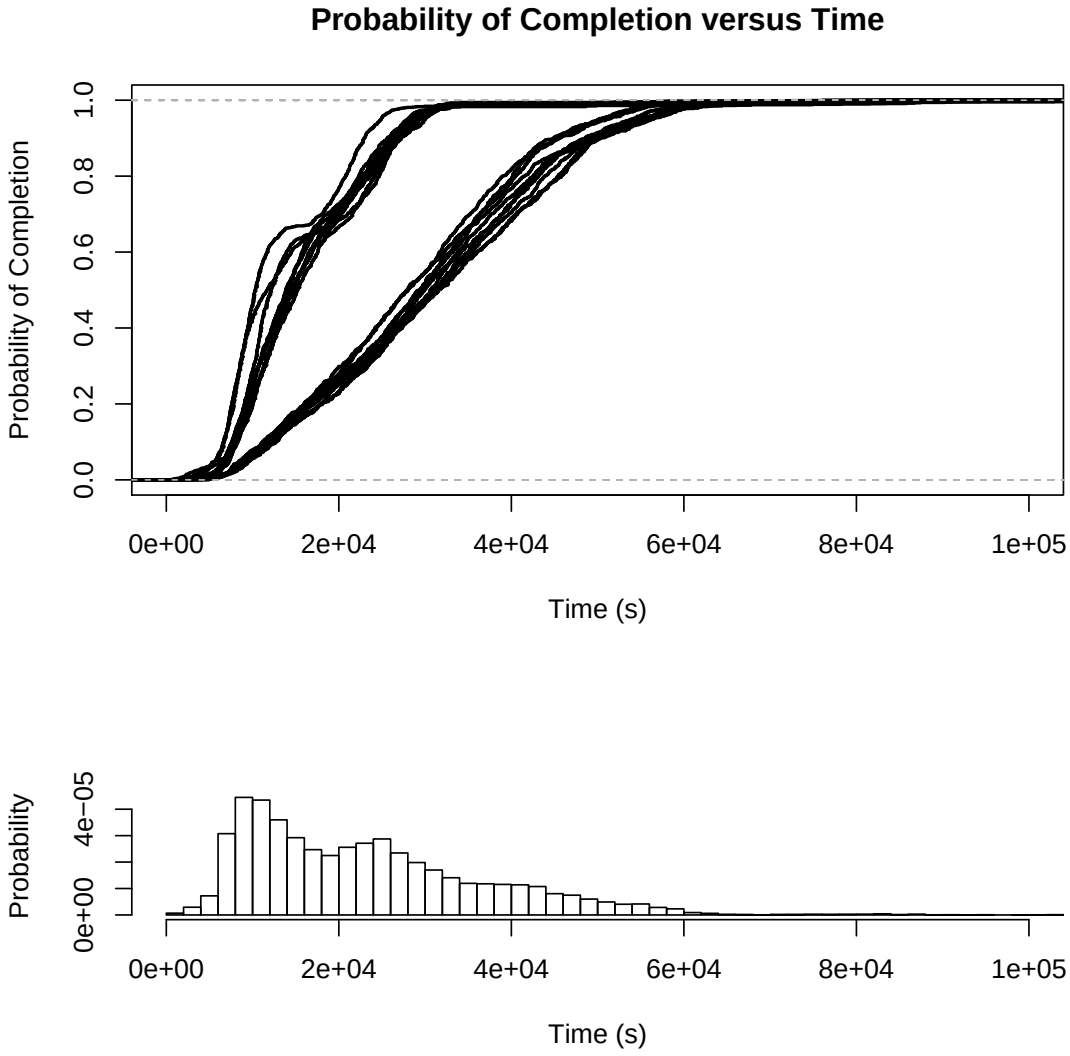

Figure 5.8 – Execution profile of jobs (WMS, type 2 tasks). The influence of the distinctive double peak distribution of the type 2 tasks can be clearly seen in both the distribution function and histogram.

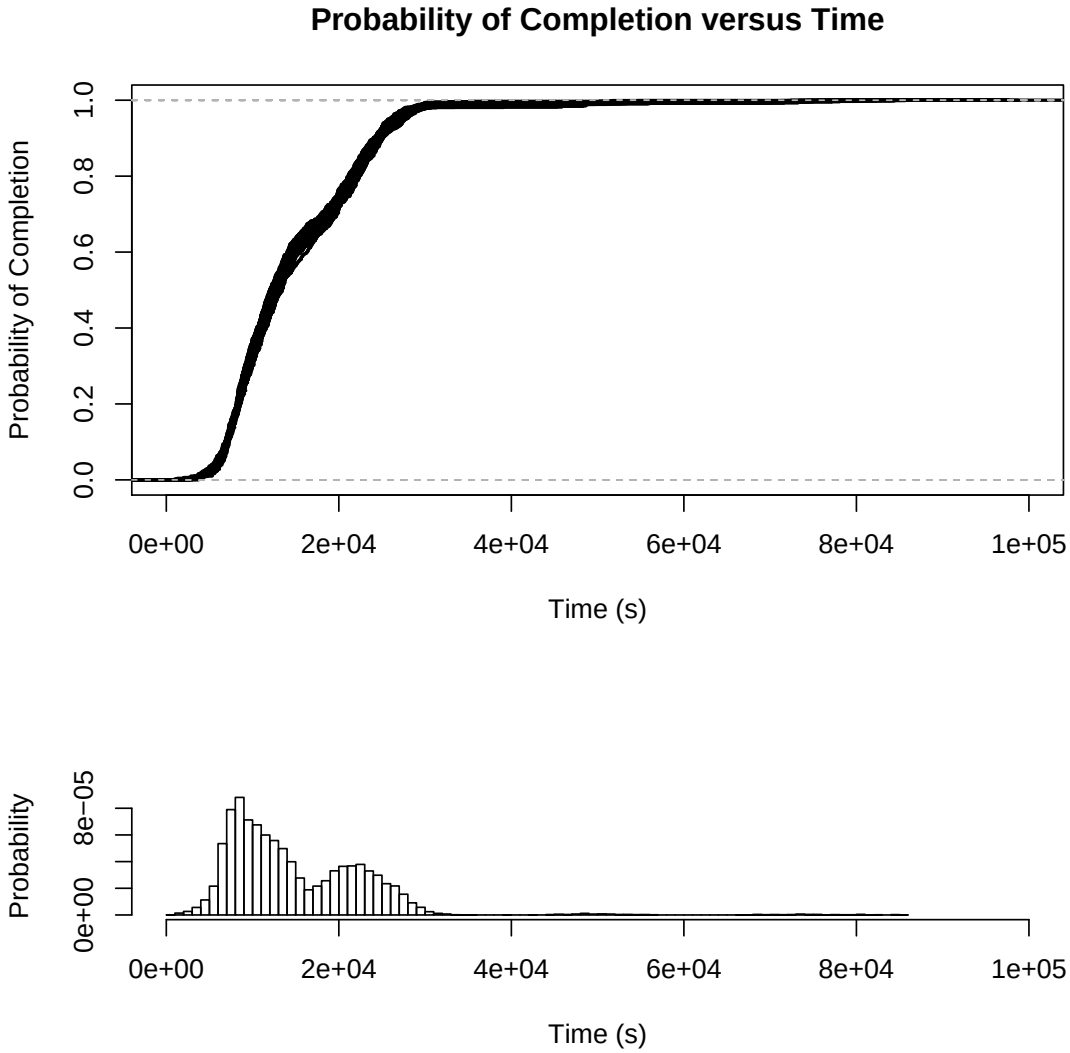

Figure 5.9 – Execution profile of jobs (IRB, type 2 tasks). The selection of the faster resources, and the distribution of tasks to multiple resources, leads to a tighter distribution of observed task durations than in *Figure 5.8*.

5.3.3 Licensed Applications

Current Operation

Due to the more-or-less random assignment of jobs to resources, and the fact that there is little consideration given to the matter of ensuring that requisite software licences are available prior to executing a job, licence requirements can have a detrimental effect on the success rate of successful job execution, particularly in more complicated cases.

For the purposes of the first tests, the test harness was configured as shown in *Table 5.6*. Where a resource is said to have licences available, there were sufficient licences for each slot in that resource to consume one concurrently. This means that, in effect, a site licence was in force, permitting all execution slots on the resource to run licensed applications at the same time.

Table 5.6 – Availability of licences on execution resources.

Resource	Licences
conan.elec.gla.ac.uk	Yes
miffy.elec.gla.ac.uk	Yes
ngs.leeds.ac.uk	No
ngs.ral.ac.uk	No
ngs.glasgow.ac.uk	Yes
ngs.manchester.ac.uk	Yes
ngs.lancaster.ac.uk	No
ngs.oxford.ac.uk	No
ngs.wmin.ac.uk	No
scotgrid.ac.uk	Yes

The outcome of running two series of simulations within this environment is presented in *Table E.5* (1,000 type 1 tasks) and *Table E.6* (1,000 type 2 tasks). Although those jobs which were sent to appropriate resources completed in the normal manner, those which were sent to inappropriate resources—those where the necessary licences were not available—failed, as would be expected. In the shorter (type 1) simulation, sixty percent of the jobs were dispatched to unsuitable resources and failed as a result; in the longer (type 2) simulation, this number was fifty percent. Regardless of the exact percentage, this demonstrates that the need to consider licence restrictions cannot be overlooked.

Improved Resource Broker

In this particular set of circumstances, the improved resource broker provides several options for submission. Assuming an affinity of 0.0—in other words, that there is no par-

ticular imperative that the tasks be executed on the same resource—the preferred option is to divide the jobs between the fastest two resources with the necessary licences (which happen to be the two fastest resources).

From the results in *Table E.8* (1,000 type 1 tasks), it can be seen that a resource with sufficient licences is always chosen, avoiding the trivial but intensely annoying situation where jobs fail due to a lack of licences. *Figure 5.10* illustrates the change in task duration when the WMS is compared with the improved resource broker, while *Figure 5.11* and *Figure 5.12* demonstrate the proportion of tasks which will have completed as a function of time. From these, it can be seen that the tasks follow a much more predictable course when the improved submission profile is used. It is also evident that the improved resource broker does not always obtain the quickest results, although on average it performs significantly better than the WMS. (This slightly counterintuitive result can be attributed to the simple estimate of queue time used by the schedule evaluator; although it bases its allocation on an estimate of mean queue time for each slot, when the number of slots greatly exceeds the number of tasks, there is always the possibility that a large proportion of those tasks executing at the time of submission will complete ahead of the predicted schedule. In this particular set of circumstances, the resource will execute the submitted tasks in a shorter time than expected.)

The results for a similar simulation (1,000 type 2 tasks) using the longer jobs are presented in *Table E.9*, and are further illustrated in *Figure 5.13*, *Figure 5.14* and *Figure 5.15*.

Table 5.7 – Job outcome: licensing (type 1 tasks).

	WMS	**IRB**
Completed	40%	100%
Failed due to missing licence	60%	0%

Table 5.8 – Job outcome: licensing (type 2 tasks).

	WMS	**IRB**
Completed	50%	100%
Failed due to missing licence	50%	0%

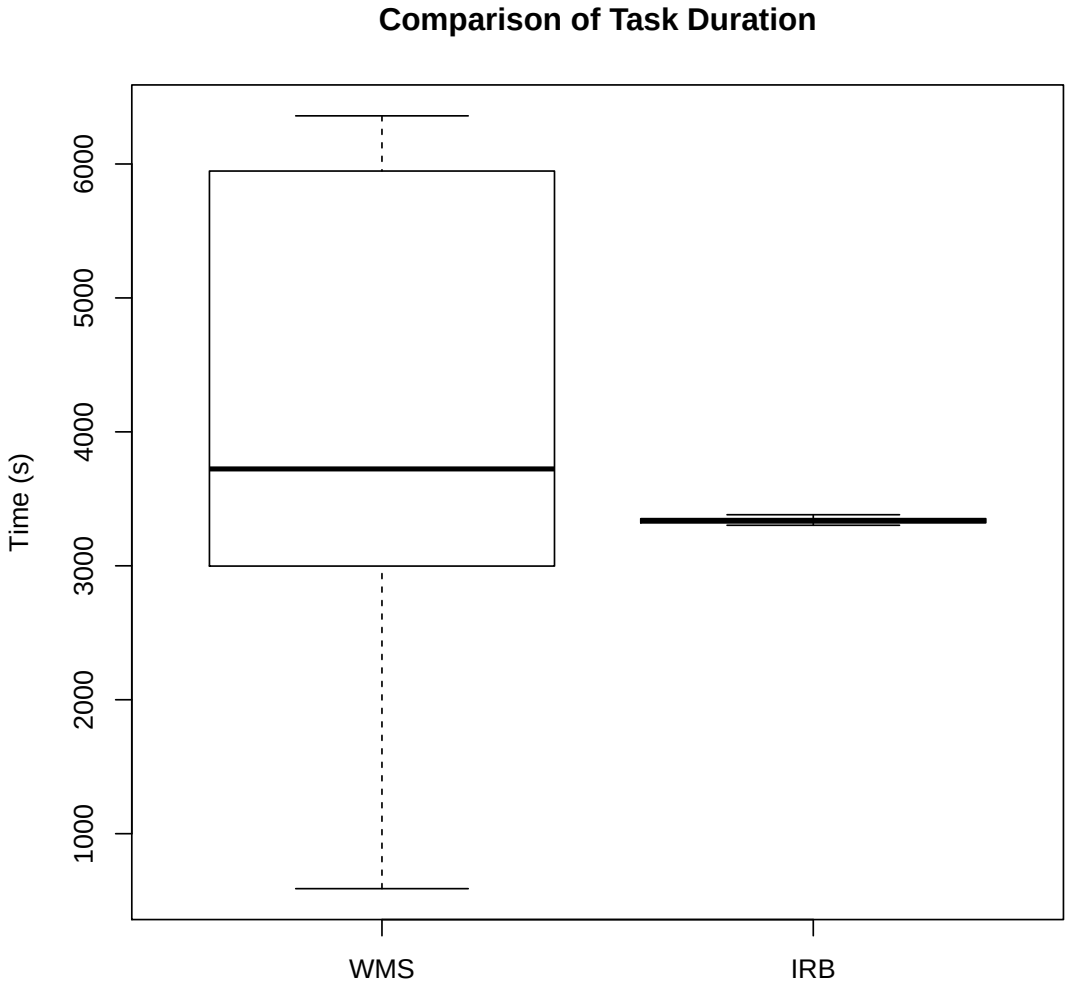

Figure 5.10 – Comparison of duration of jobs with licence (type 1 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

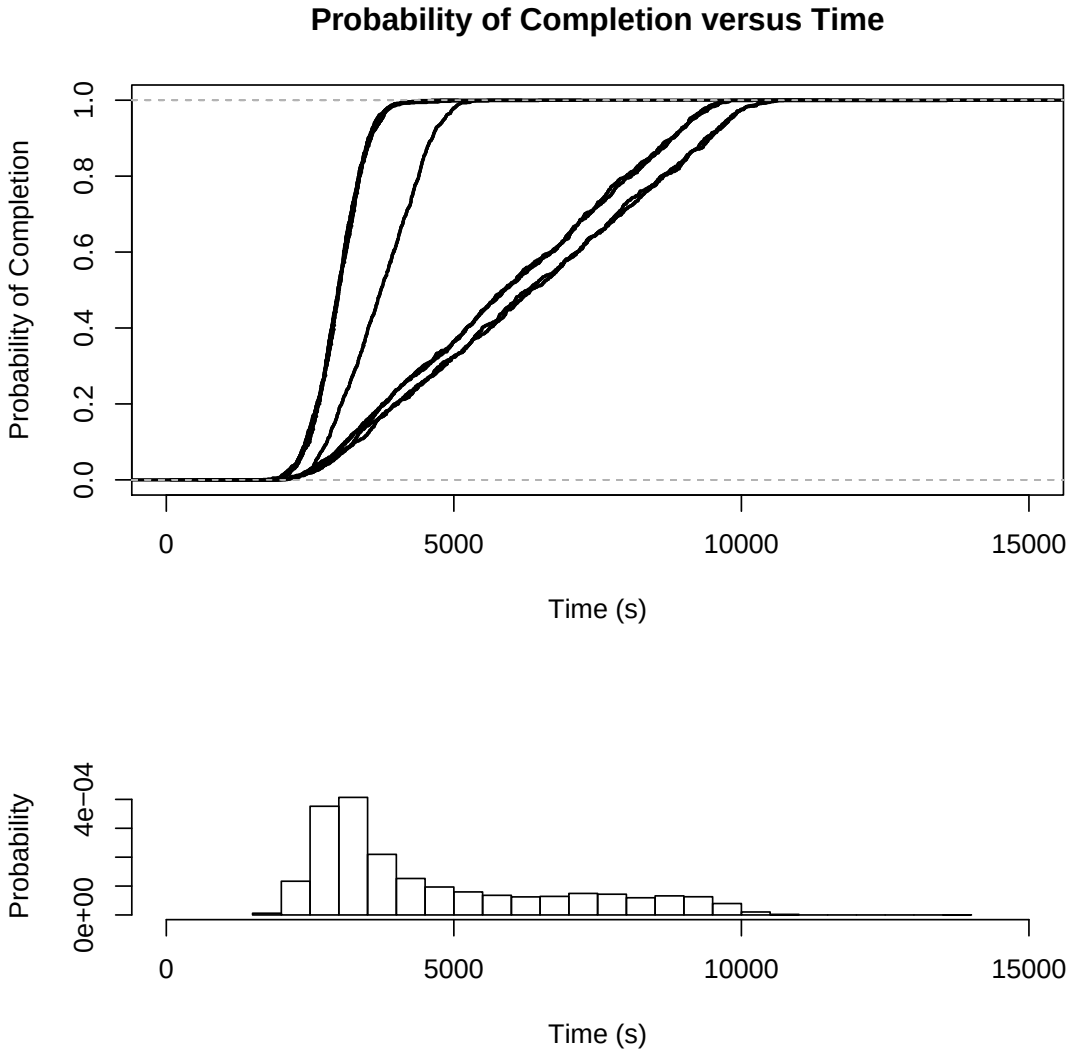

Figure 5.11 – Execution profile of jobs with licence (WMS, type 1 tasks). Tasks which failed due to licences not being available are not shown, which accounts for the reduced data in the graphs.

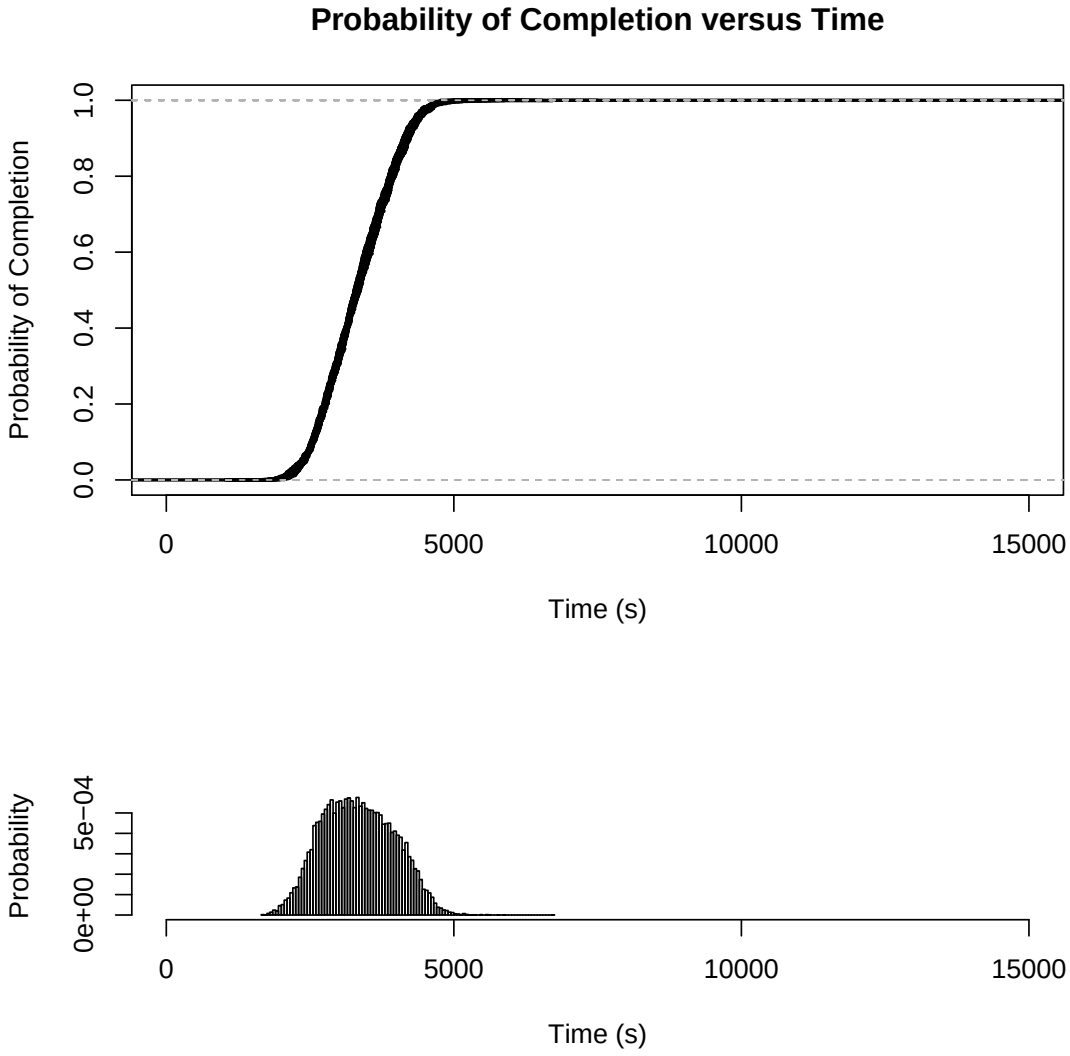

Figure 5.12 – Execution profile of jobs with licence (IRB, type 1 tasks). Licence requirements were satisfied in all cases, and all tasks completed successfully.

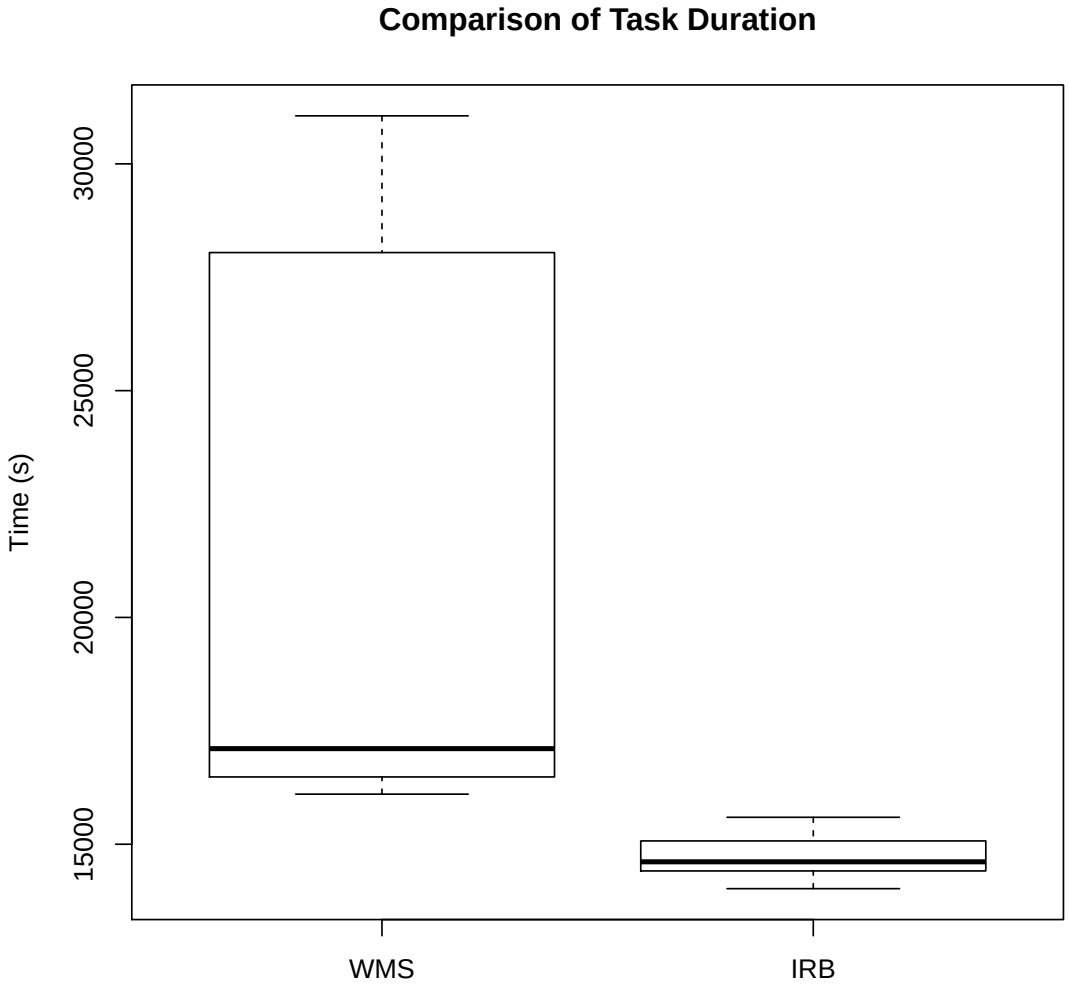

Figure 5.13 – Comparison of duration of jobs with licence (type 2 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

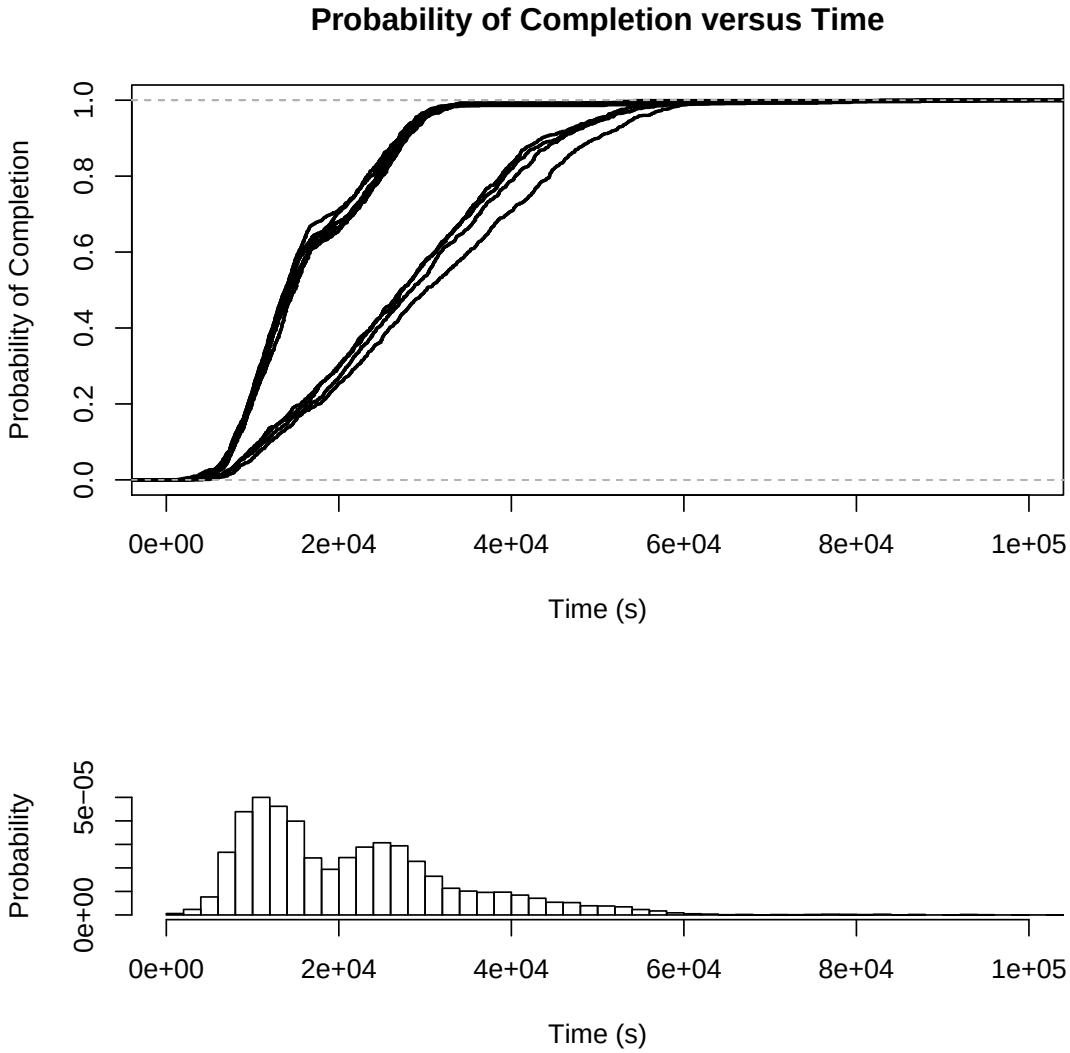

Figure 5.14 – Execution profile of jobs with licence (WMS, type 2 tasks). Tasks which failed due to licences not being available are not shown, which accounts for the reduced data in the graphs.

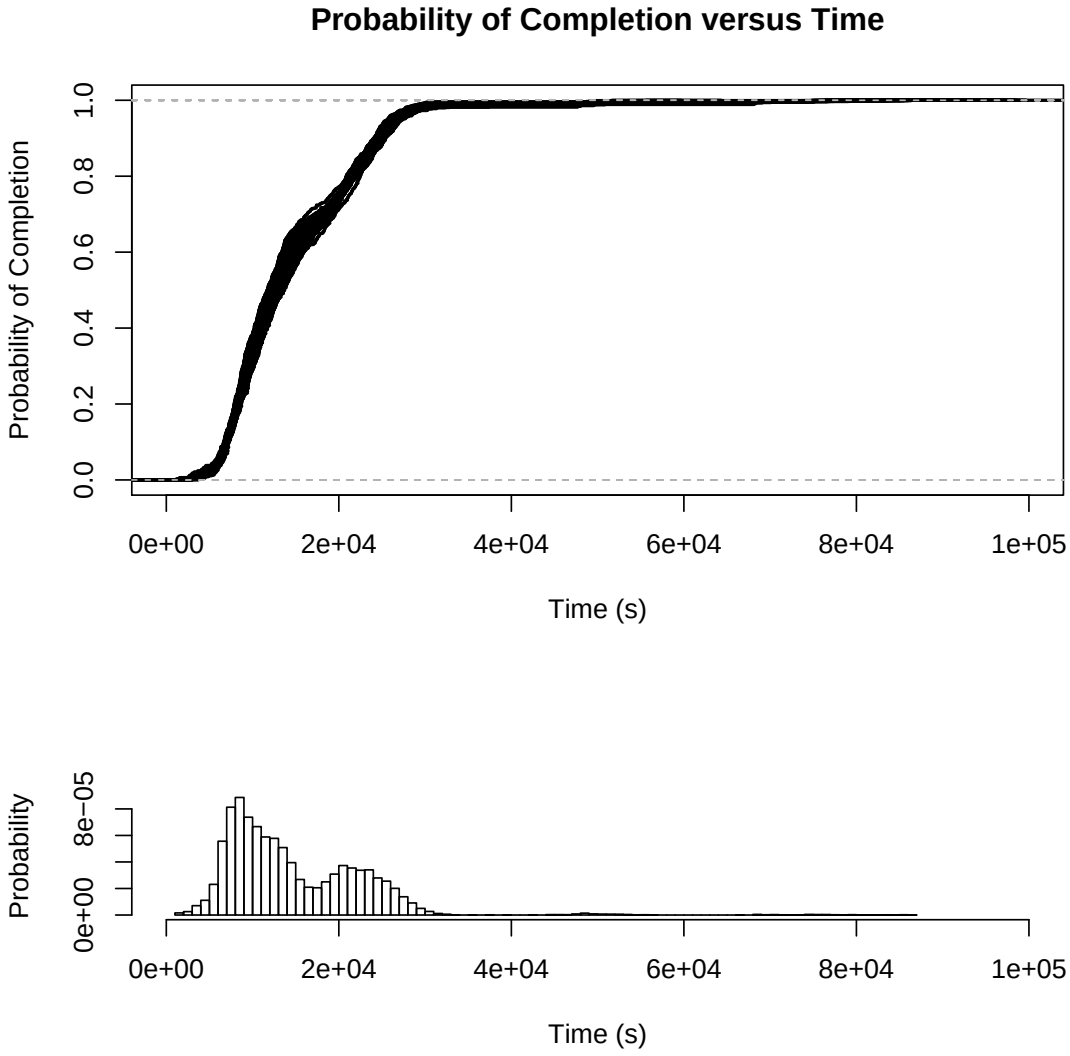

Figure 5.15 – Execution profile of jobs with licence (IRB, type 2 tasks). Licence requirements were satisfied in all cases, and all tasks completed successfully.

In a slightly more complicated situation, where the number of licences held by a resource is less than the total number of execution slots, the improved resource broker will distribute jobs in such a way as to avoid exceeding the quota held by any one resource. For example, given the licence distribution presented in *Table 5.9*, the results detailed in *Table E.7* and *Table E.10* are obtained; these are illustrated in *Figure 5.16*, *Figure 5.17* and *Figure 5.18*. As with the previous case, with the improved resource broker, all jobs completed successfully in this situation. This time, a reduction in mean duration of 1,496 seconds (32.5%) was observed.

These results demonstrate a significant improvement in the successful completion of tasks when information about the licence requirements of jobs is taking into consideration during the allocation of work to execution resources. Although these particular tests have not made use of personal licences, information about personal licences is used by the improved resource broker when it prepares allocations of tasks to resources, either to supplement licences provided by other means, or to bypass the requirement altogether (in cases where the user has an unlimited supply of licences, either permanently or for a particular job) by acting as if the number of licences available on a resource is equal to the number of execution slots offered by the resource. In such a way, licensing models where the licence is attached to a particular set of data, and which have been proposed as one approach to resolving some of the issues affecting software licensing in the distributed computing domain [51], could be supported.

Many resources will maintain their own count of licence availability, independent of systems provided through any grid computing infrastructure. This permits local users, who may be able to run jobs without resorting to the use of elaborate grid tools, to make use of the same licensed software as remote users. Ensuring that licence information is specified in a clear and open manner, and incorporating licence information within the improved resource broker in a flexible, modular fashion, should help ease the process of integrating local licence management strategies with the improved resource broker. Further discussion of the problems associated with integrating grid and local job schedulers and infrastructure is provided in *Appendix C – Integration with Local Job Managers*.

Table 5.9 – Availability of licences on execution resources.

Resource	Licences
conan.elec.gla.ac.uk	300
miffy.elec.gla.ac.uk	300
ngs.leeds.ac.uk	No
ngs.ral.ac.uk	No
ngs.glasgow.ac.uk	300
ngs.manchester.ac.uk	300
ngs.lancaster.ac.uk	No
ngs.oxford.ac.uk	No
ngs.wmin.ac.uk	No
scotgrid.ac.uk	300

Table 5.10 – Job outcome: restricted licensing.

	WMS	IRB
Completed	19%	100%
Failed due to missing licence	81%	0%

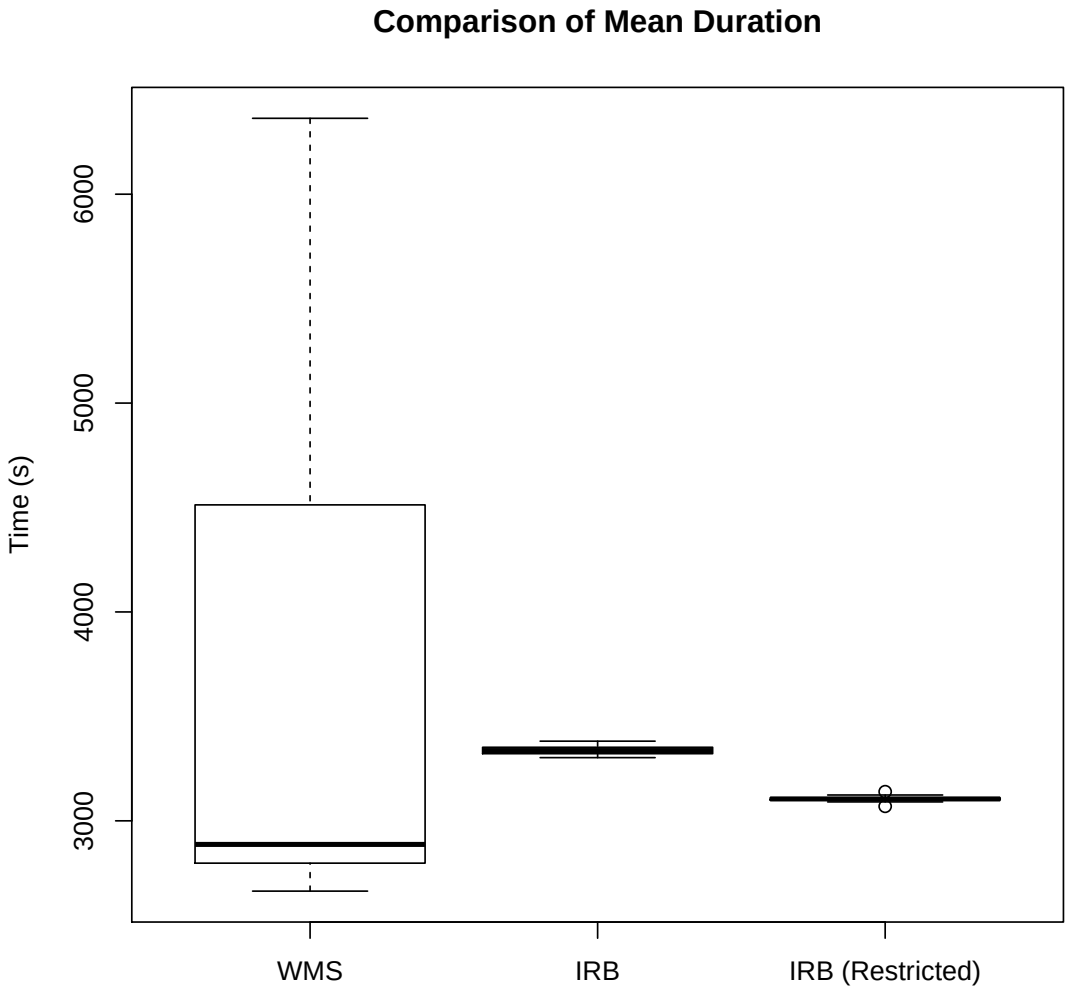

Figure 5.16 – Comparison of duration of jobs with restrictive licence (type 1 tasks), illustrating the variation in mean task duration within the simulations. Where the sample minima and maxima fall within one-and-a-half times the interquartile range, these are indicated by the lower and upper lines; otherwise, these lines indicate the minima and maxima within this range, and outliers are marked with a circle. In all cases, the boxes indicate the extent of the second and third quartiles (the interquartile range), and the solid lines denote the median values.

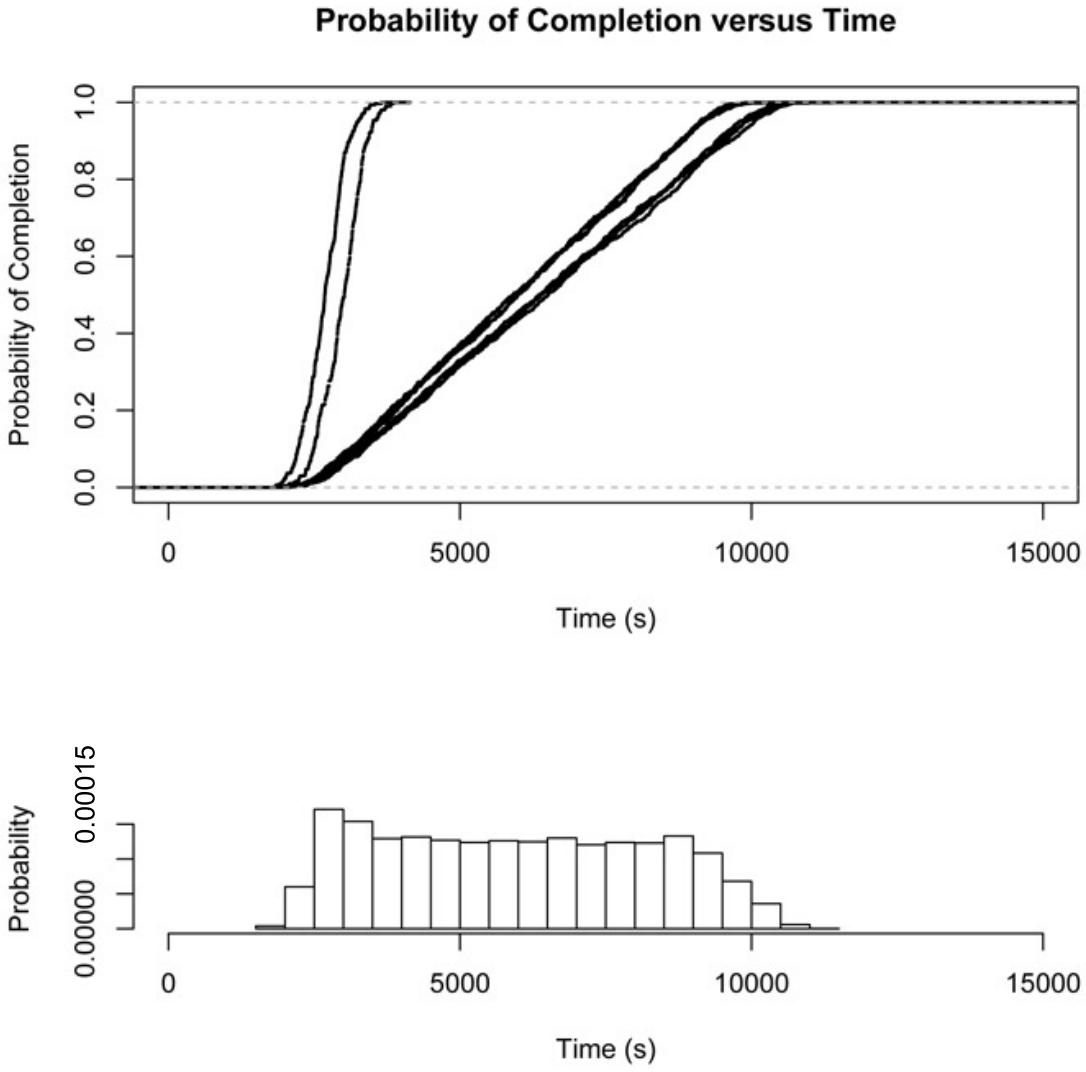

Figure 5.17 – Execution profile of jobs with restrictive licence (WMS, type 1 tasks). Fewer licences were available than the number of hosts within a resource. Tasks which failed due to licences not being available are not shown, which accounts for the reduced data in the graphs.

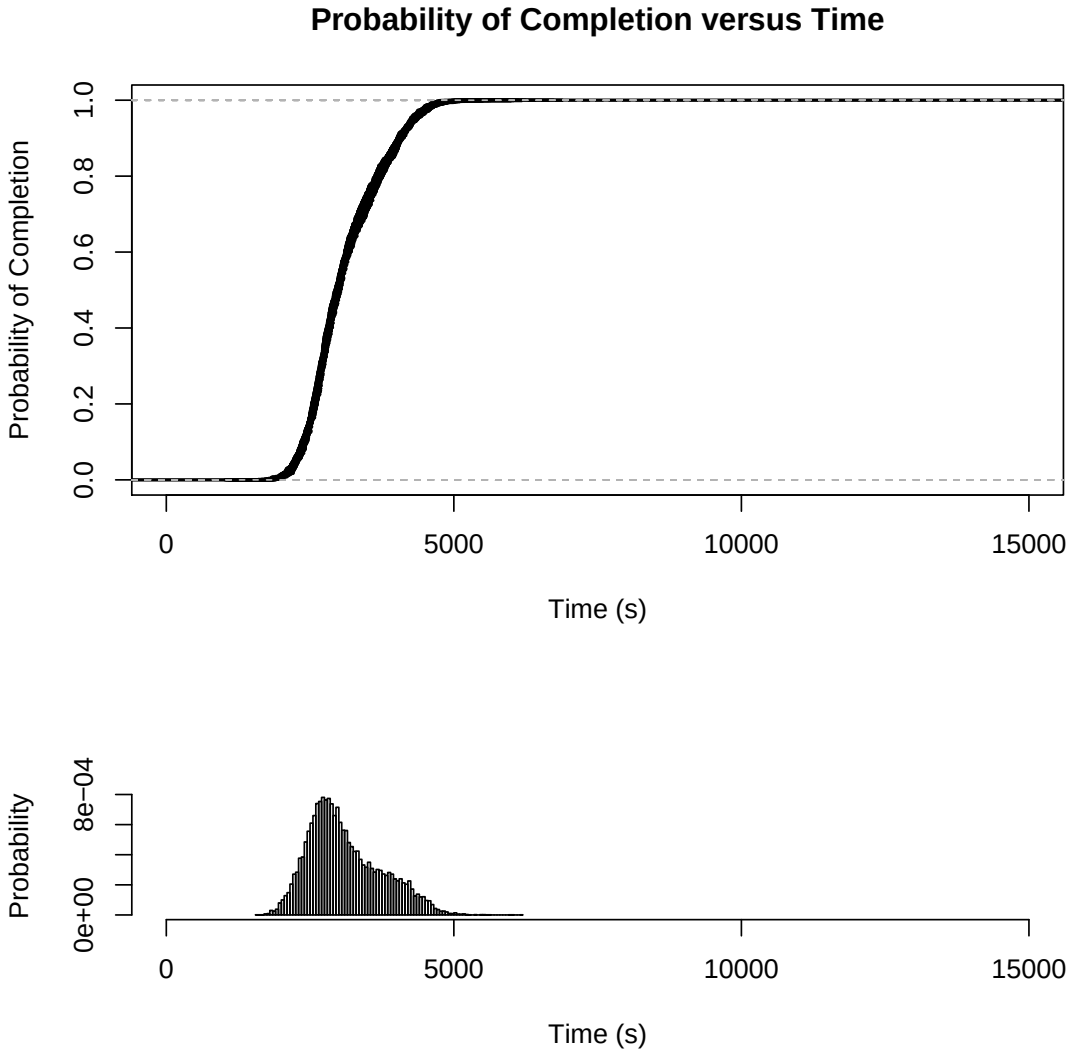

Figure 5.18 – Execution profile of jobs with restrictive licence (IRB, type 1 tasks). Fewer licences were available than the number of hosts within a resource. Licence requirements were satisfied in all cases, and all tasks completed successfully.

5.3.4 Deadlines

Current Operation

The selection of the least-loaded system, if this were to provide the tasks with the fastest route to completion, would be advantageous when trying to complete a package of work before a deadline. However, the restriction to a single resource greatly limits the potential options with regards to allocation, and makes overall success less likely.

For the purposes of this test, an arbitrary deadline was assumed to be in effect. This was two hours thirty minutes for the short example, and twelve hours for the longer example, the time period commencing at the moment of submission. Summarized data is presented in *Table E.11* (1,000 type 1 tasks) and *Table E.12* (1,000 type 2 tasks).

Improved Resource Broker

Once again, the improved resource broker provides several options for submission with this set of constraints. Assuming an affinity of 0.0—in other words, that there is no particular imperative that the tasks be executed on the same resource—the first option is to apportion tasks among the fastest resources. Summarized data is presented in *Table E.13* (1,000 type 1 tasks) and *Table E.14* (1,000 type 2 tasks). A comparison of the success rates of type 1 tasks (*Table 5.11*) and type 2 tasks (*Table 5.12*) illustrates observed behaviour in these situations.

From *Figure 5.19*, it can be seen that there is a reduction in processing time when use is made of the improved resource broker, and from *Table 5.12*, it can be seen that the average success rate of 98.8% represents a significant improvement over the previous case (87.8%). While the average execution time of a task in this example was approximately three hours, the presence of occasional tasks well in excess of this duration means that the ideal one hundred percent success rate is unobtainable in this particular set of circumstances.

Table 5.11 – Job outcome (type 1 tasks with deadline).

	WMS	**IRB**
Completed	90.5%	100%
Missed deadline	9.5%	0%

Table 5.12 – Job outcome (type 2 tasks with deadline).

	WMS	IRB
Completed	87.8%	98.8%
Missed deadline	12.2%	1.2%

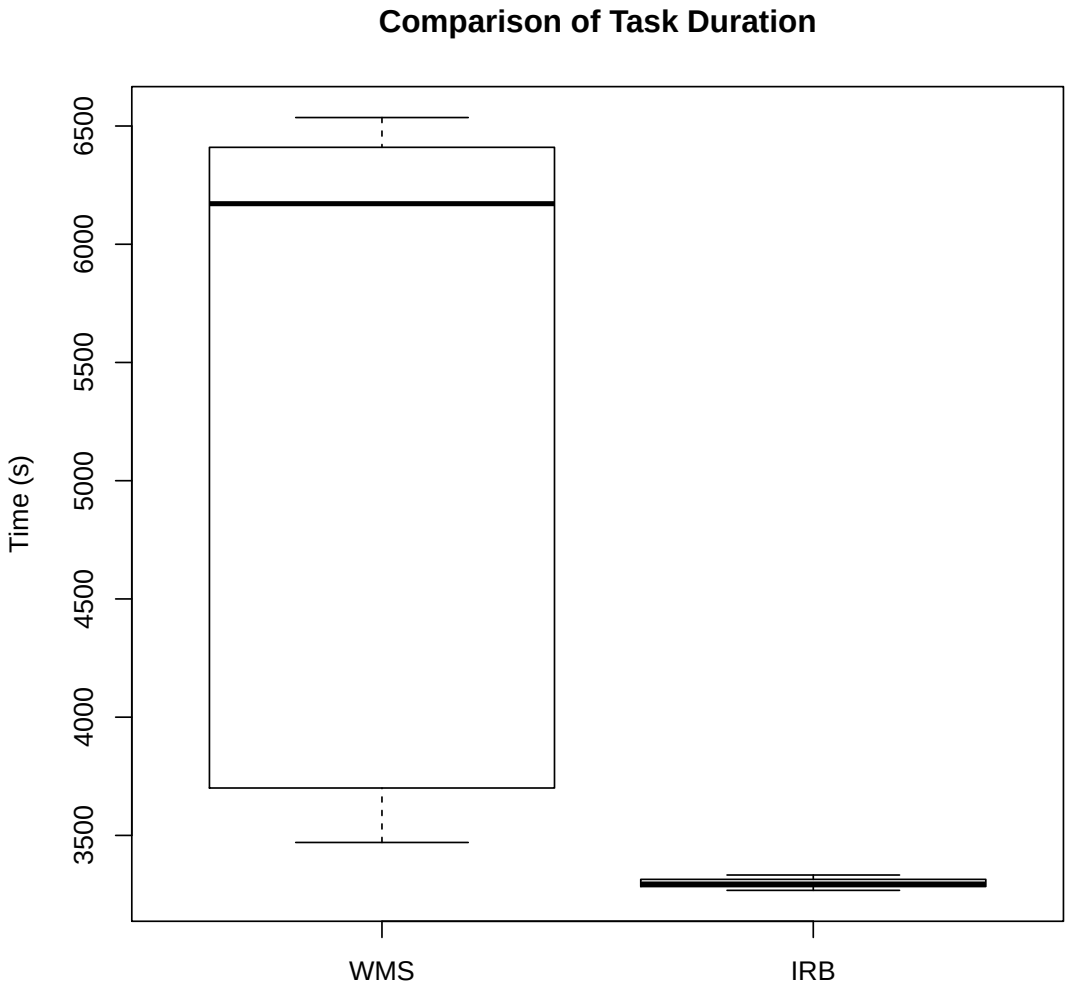

Figure 5.19 – Comparison of duration of jobs with deadline (type 1 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values. In neither case does the mean duration exceed the deadline (9,000 seconds in this example).

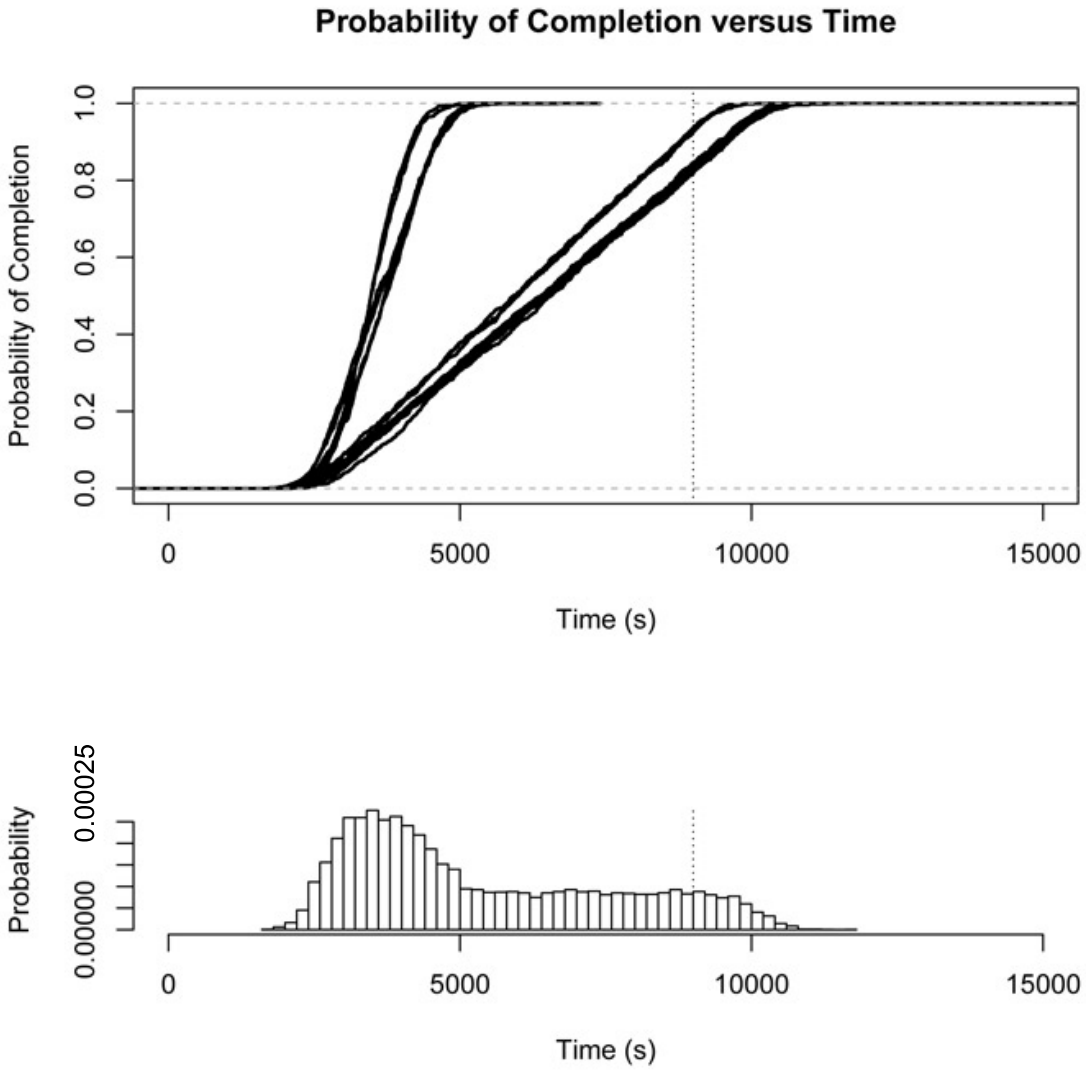

Figure 5.20 – Execution profile of jobs with deadline (WMS, type 1 tasks). The dotted line marks the deadline of 9,000 seconds. As the resource selection does not take this into account, it can be seen that resources are sometimes selected where, in this particular example, up to 20% of tasks are unlikely to have finished prior to the deadline.

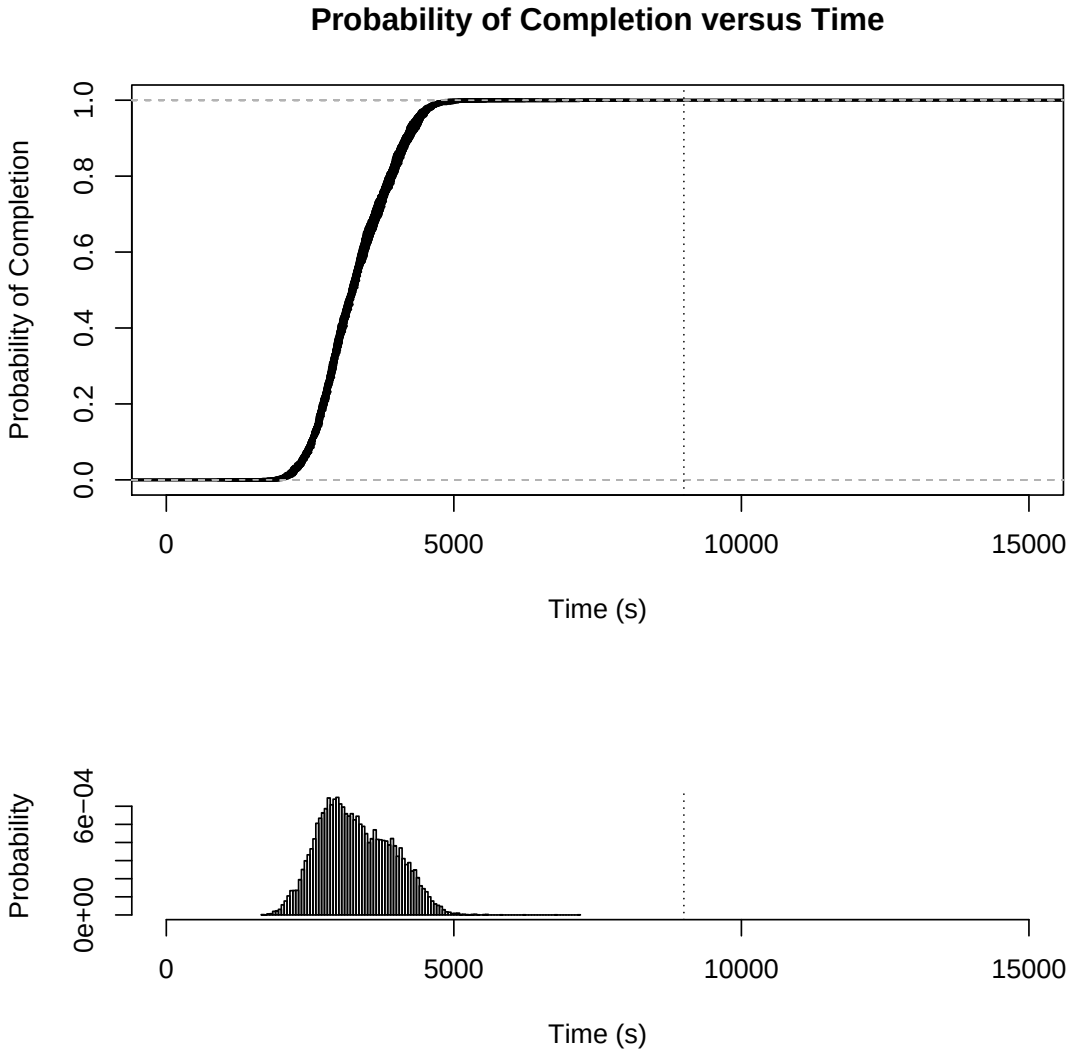

Figure 5.21 – Execution profile of jobs with deadline (IRB, type 1 tasks). The dotted line marks the deadline of 9,000 seconds. Tasks have been distributed to resources in order to maximize the likelihood of completion prior to the deadline.

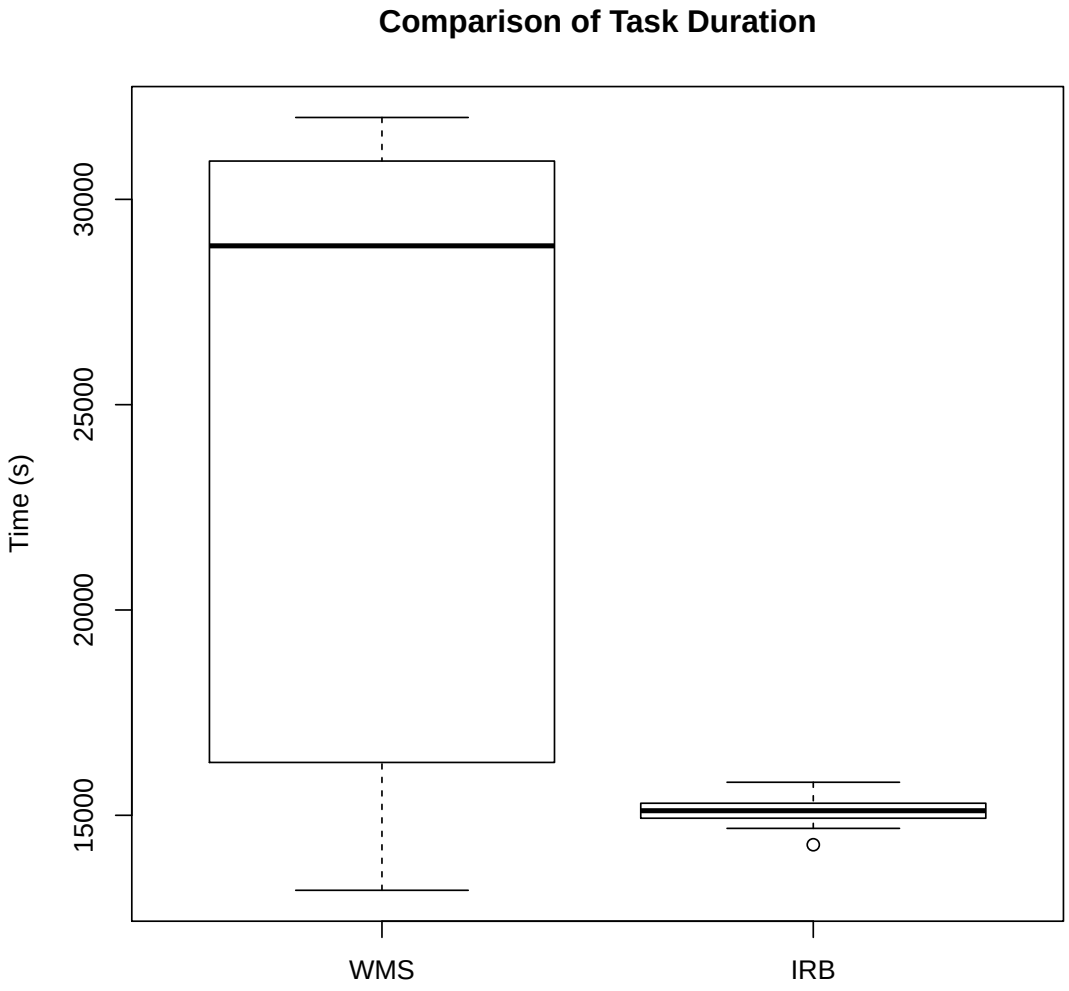

Figure 5.22 – Comparison of duration of jobs with deadline (type 2 tasks), illustrating the variation in mean task duration within the simulations. Where the sample minima and maxima fall within one-and-a-half times the interquartile range, this is indicated by the lower and upper lines; otherwise, these lines indicate the minima and maxima within this range, and outliers are marked with a circle. In both cases, the boxes indicate the extent of the second and third quartiles (the interquartile range), and the solid lines denote the median values. In neither case does the mean duration exceed the deadline (43,200 seconds in this example).

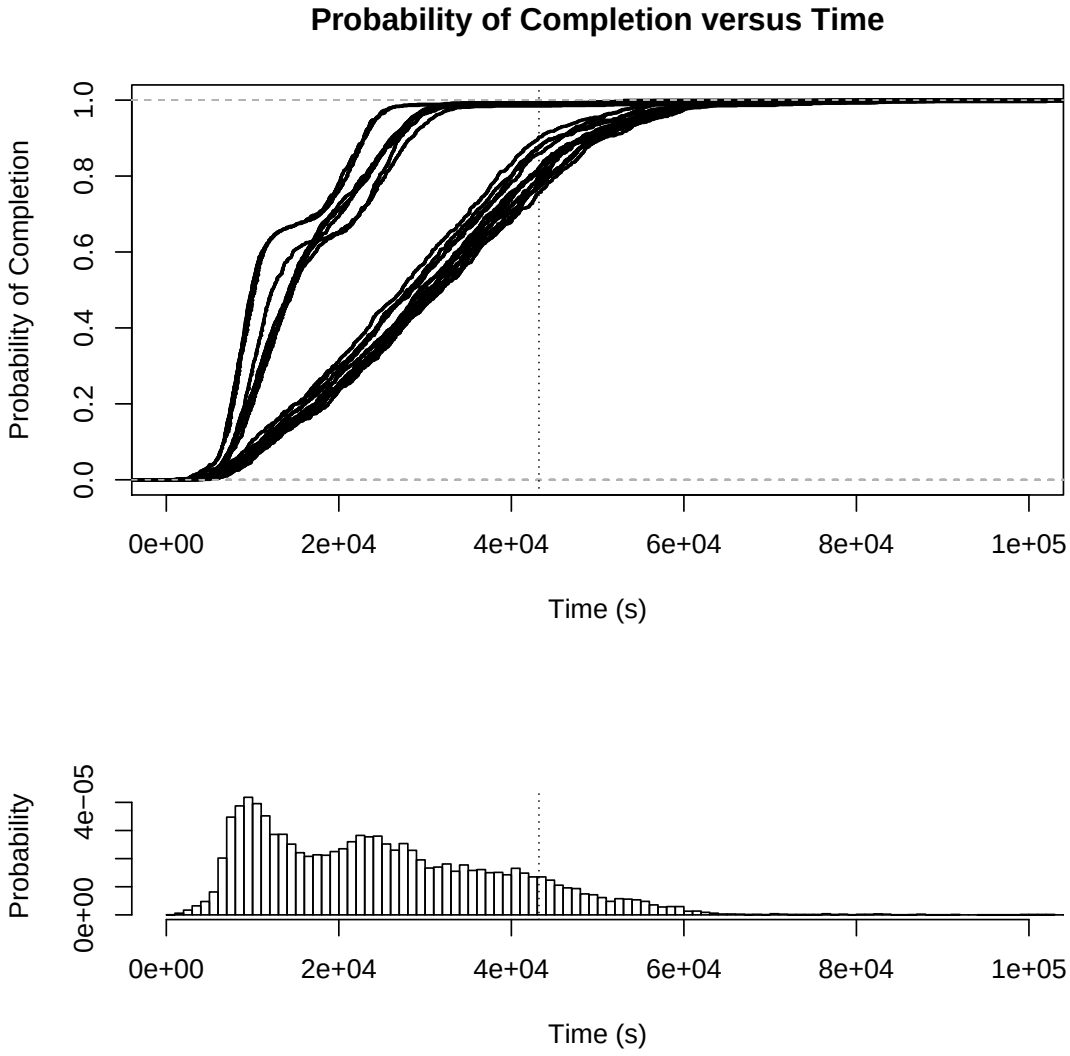

Figure 5.23 – Execution profile of jobs with deadline (WMS, type 2 tasks). The dotted line marks the deadline of 43,200 seconds. As in *Figure 5.20*, ignorance of deadline means resources are sometimes selected where a significant proportion of jobs have little chance of completing in sufficient time.

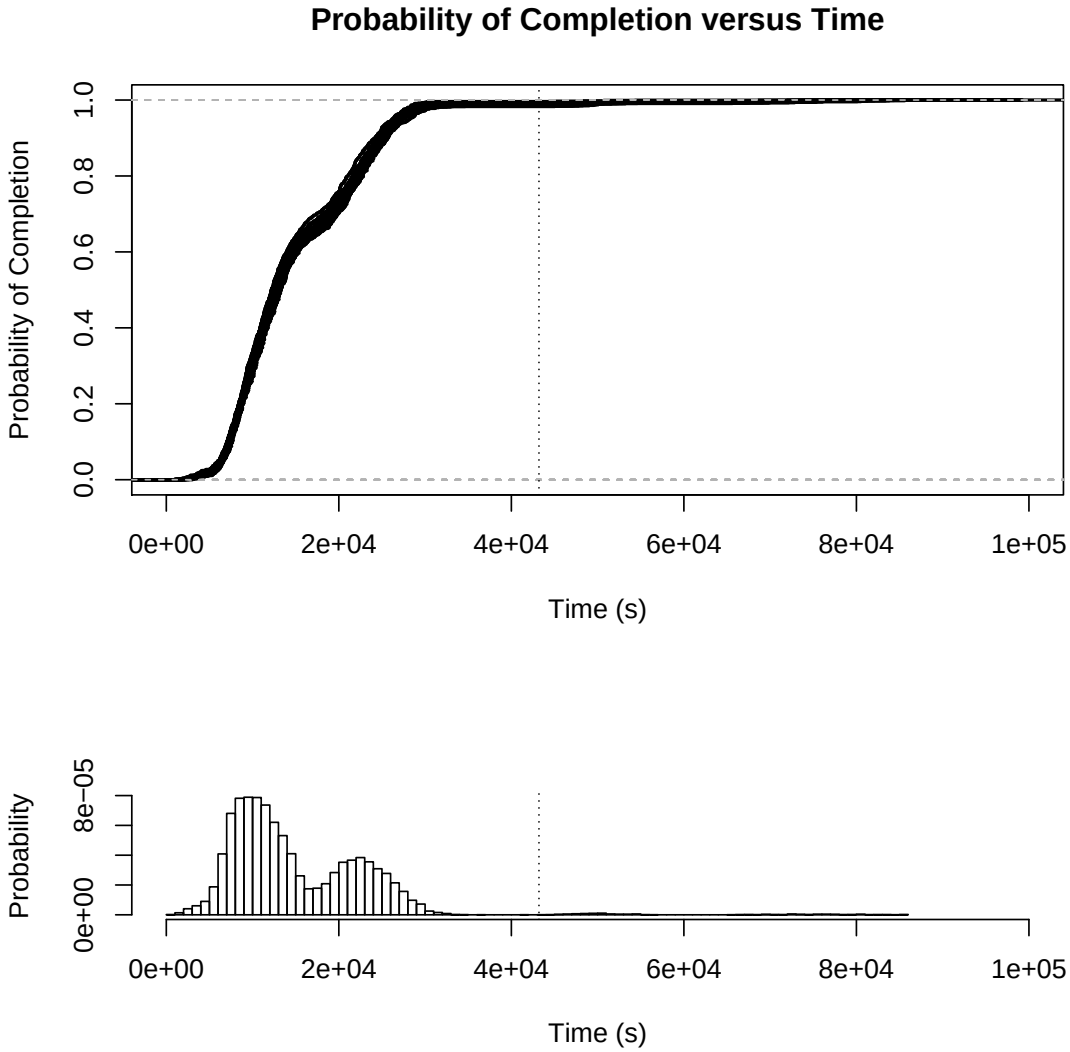

Figure 5.24 – Execution profile of jobs with deadline (IRB, type 2 tasks). The dotted line marks the deadline of 43,200 seconds. The presence of significant outliers in the distribution of task durations means that it is extremely difficult to ensure that all tasks have completed prior the nominated deadline. Nevertheless, there is a significant improvement in success rate over the previous example.

5.3.5 Downtime

Downtime is a problem that affects every computational resource. Some downtime will be unplanned: mechanical components wear out, electronic components fail, and electricity supplies are not always entirely stable (particularly those to academic research facilities which cannot justify the financial outlay necessary to secure a resilient supply). Nevertheless, managing any resource will also inevitably involve the need to resort to planned periods of downtime to undertake routine inspections and maintenance, or to perform system-wide upgrades of software or hardware. The effects of downtime of data providers can be mitigated by the use of replication (of particular importance when applying grid computing techniques to real-world problems demanding a degree of immediacy or consistency in performance, such as those related to the management and manipulation of medical imaging data [132]), and it is possible to work around periods of downtime by removing resource listings temporarily from grid information systems [65]; procedures are also generally in place to require dissemination of information regarding the cause of the downtime, by e-mail or other means [151]. The improved resource broker builds on this position by enabling a resource to advertise a period of planned downtime well in advance, but to continue doing useful work up until this period begins.

Current Operation

Presently, no consideration is given to scheduled downtime of execution resources, despite the fact that such periods are relatively common occurrences. Although the effect a period of downtime has on an application will depend very much on the application itself—one which checkpoints regularly and has no deadline will suffer fewer ill effects than one which simply crashes without returning any sort of useful result—ideally this would be taken into account when determining which resources might be suitable on which to run a job. As has been previously discussed, random downtime caused by such problems as hardware failures is not catered for, as it is impossible (or, in any case, extremely difficult) to predict when it might occur.

For the purposes of the first tests, the test harness was configured as shown in *Table 5.13*. A listed downtime of +02.00 Hours, for example, indicates that the period for which the resource is unavailable begins two hours after the simulation commences. Where a range of times is given, the resource is deemed to become available again at the second indicated time. This is one area for which the use of simulation is invaluable, as it allows tests with periods of disruption to be undertaken without the need to interrupt service on live resources.

Table 5.13 – Downtime scheduled on execution resources (type 1 tasks).

Resource	Downtime
conan.elec.gla.ac.uk	+1.75 Hours
miffy.elec.gla.ac.uk	No
ngs.glasgow.ac.uk	No
ngs.lancaster.ac.uk	+7.50 Hours
ngs.leeds.ac.uk	No
ngs.manchester.ac.uk	No
ngs.oxford.ac.uk	+1.50 Hours
ngs.ral.ac.uk	+0.50 Hours
ngs.wmin.ac.uk	No
scotgrid.ac.uk	00.00 Hours – +01.00 Hours

Table 5.14 – Downtime scheduled on execution resources (type 2 tasks).

Resource	Downtime
conan.elec.gla.ac.uk	+12.00 Hours
miffy.elec.gla.ac.uk	No
ngs.glasgow.ac.uk	No
ngs.lancaster.ac.uk	+10.00 Hours
ngs.leeds.ac.uk	No
ngs.manchester.ac.uk	No
ngs.oxford.ac.uk	+03.00 Hours
ngs.ral.ac.uk	+08.00 Hours
ngs.wmin.ac.uk	No
scotgrid.ac.uk	00.00 Hours – +02.00 Hours

Improved Resource Broker

In this situation, the selection of resources is not only based on certain performance metrics, but also makes an attempt to estimate when a task might execute on a given resource, and to ensure that this does not intersect with a predefined period of downtime.

The success rate for a series of simulations using optimistic estimates of the duration of constituent tasks is shown in *Table 5.15* (extended summaries are provided in *Table E.15* and *Table E.17*). Even when estimating the task duration optimistically—i.e. choosing a duration such that a significant proportion of tasks will likely have an actual duration in excess of the estimate—the improvement is marked, with an improvement in the overall success rate from 86.7% to over 99.0% being observed. Illustrations of behaviour are provided in *Figure 5.25*, *Figure 5.26* and *Figure 5.27*.

The success rate when the simulation is repeated with type 2 (long) tasks is shown in *Table 5.16* (extended summaries are provided in *Table E.16* and *Table E.19*), with behaviour illustrated by *Figure 5.28*, *Figure 5.29* and *Figure 5.30*. As with the shorter example, the ideal scenario of one hundred percent completion is not achieved, but it is nevertheless clear that a significant reduction in the failure rate—from 17.8% down to 1.0%—is obtained.

Table 5.15 – Job outcome: downtime (type 1 tasks).

	WMS	**IRB**
Completed	86.7%	99.5%
Failed due to downtime	13.3%	0.5%

Table 5.16 – Job outcome: downtime (type 2 tasks).

	WMS	**IRB**
Completed	82.2%	99.0%
Failed due to downtime	17.8%	1.0%

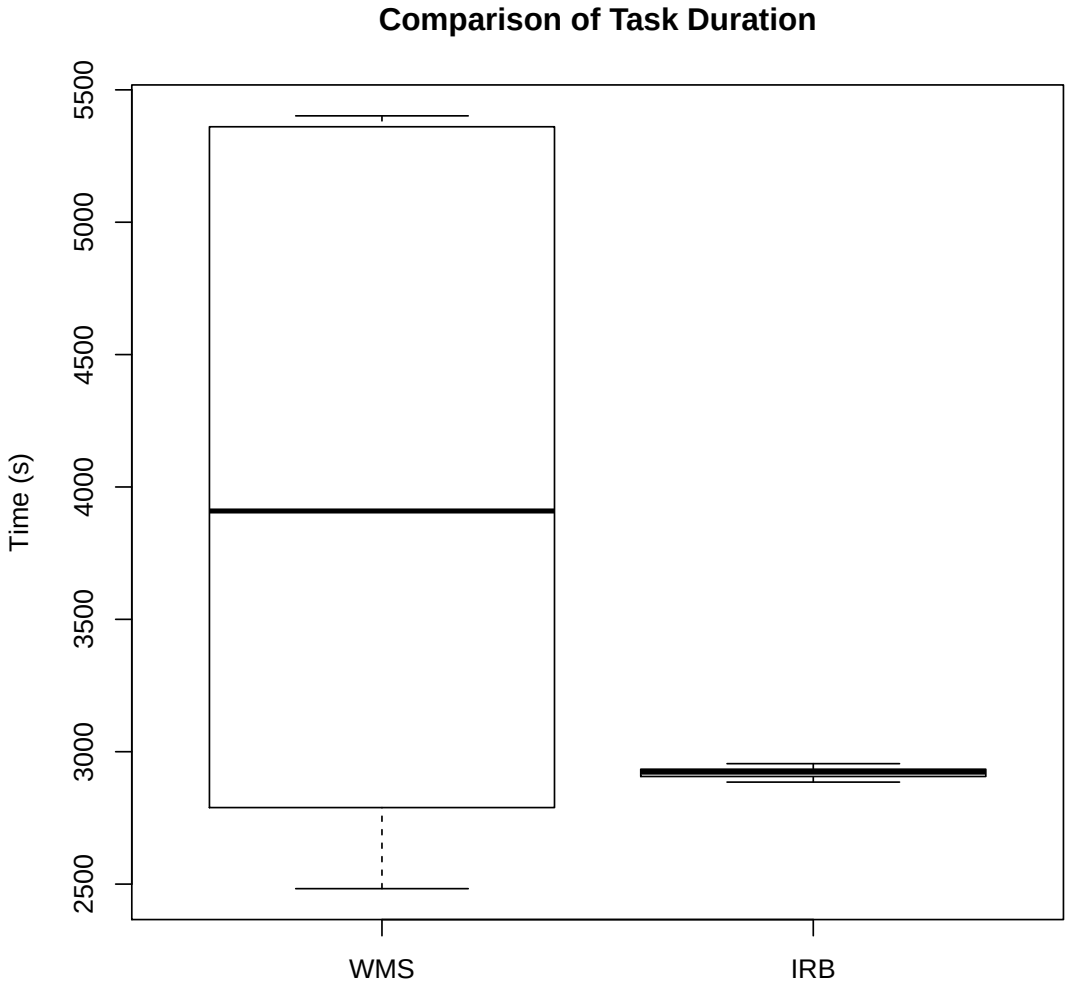

Figure 5.25 – Comparison of duration of jobs with downtime (type 1 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

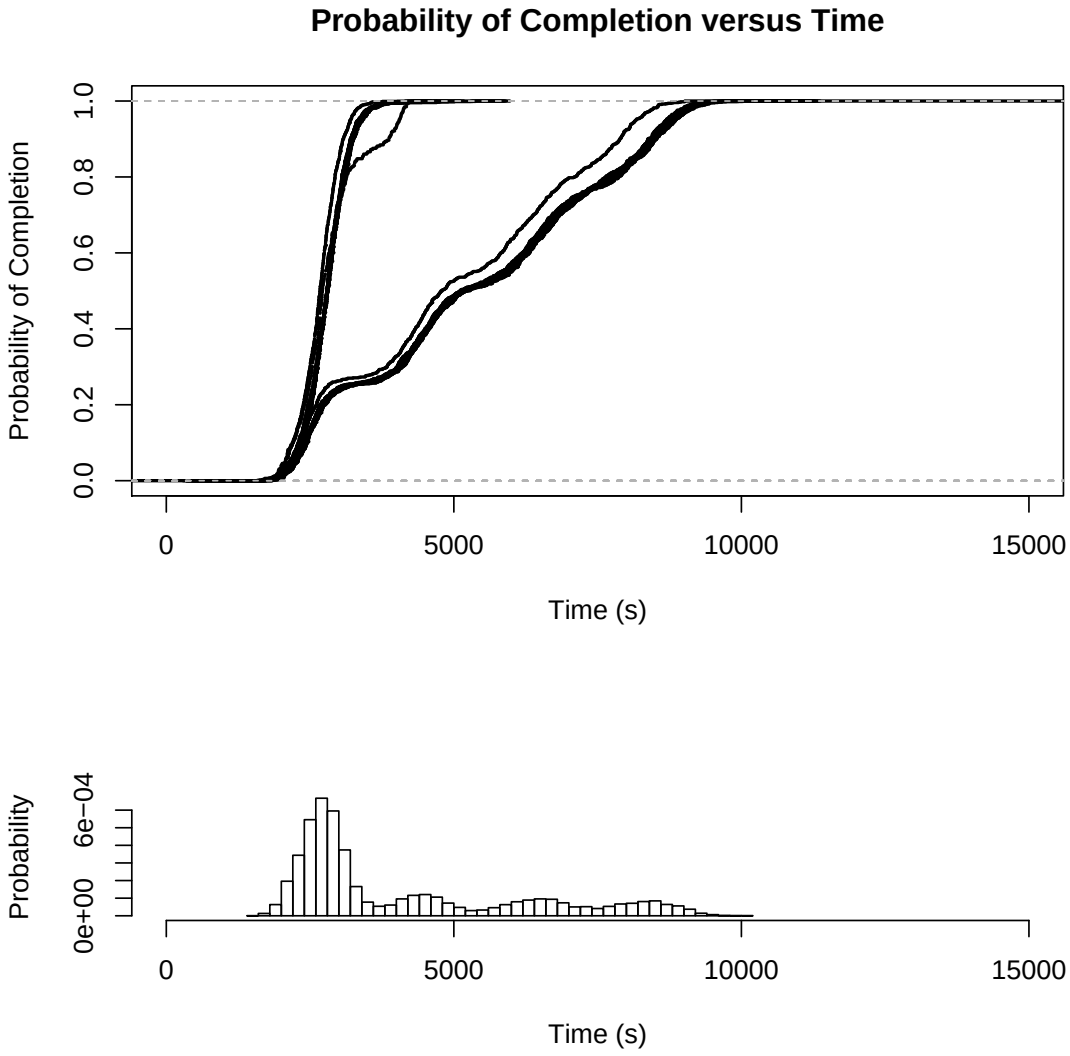

Figure 5.26 – Execution profile of jobs with downtime (WMS, type 1 tasks). Tasks which failed due to the effects of downtime are not shown.

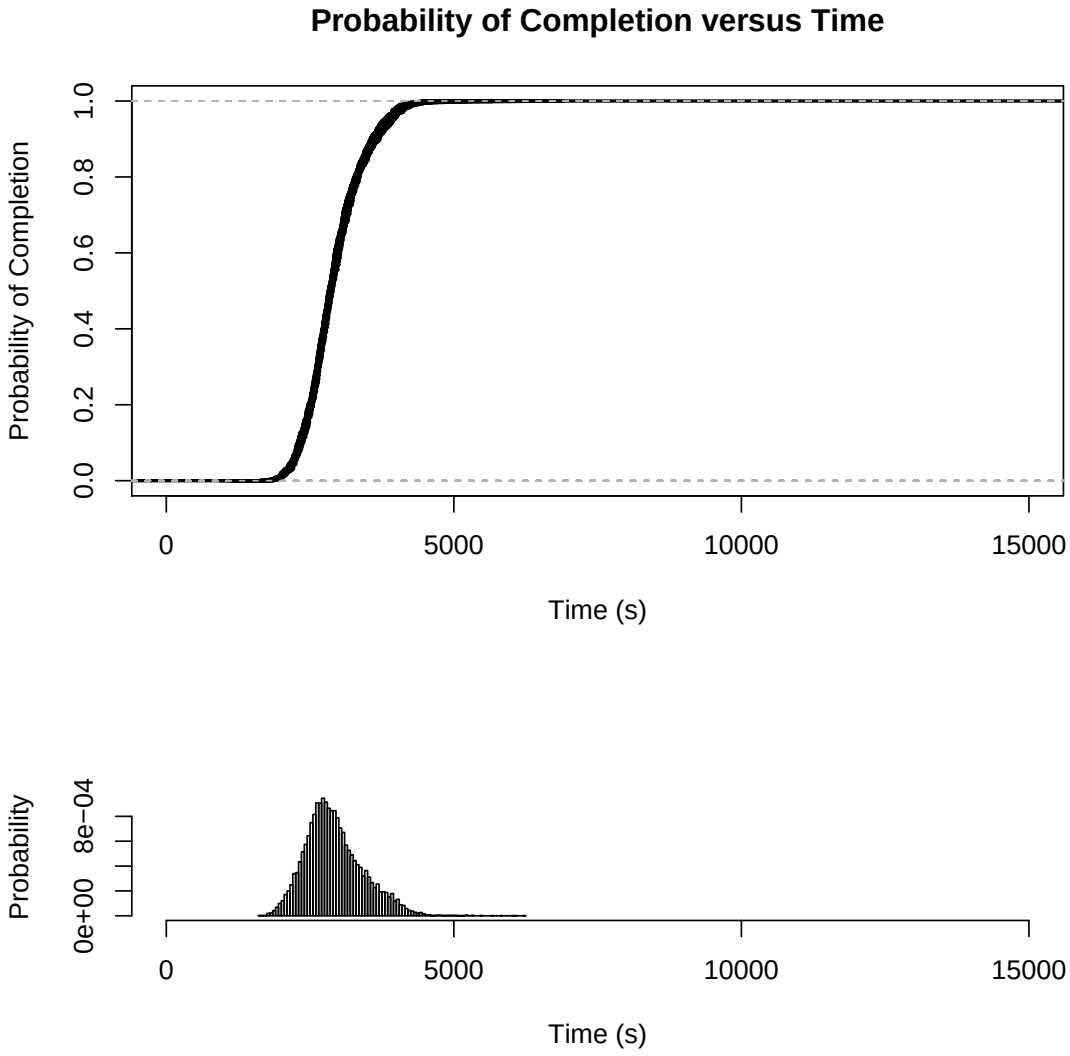

Figure 5.27 – Execution profile of jobs with downtime (IRB, type 1 tasks). Tasks which failed due to the effects of downtime are not shown.

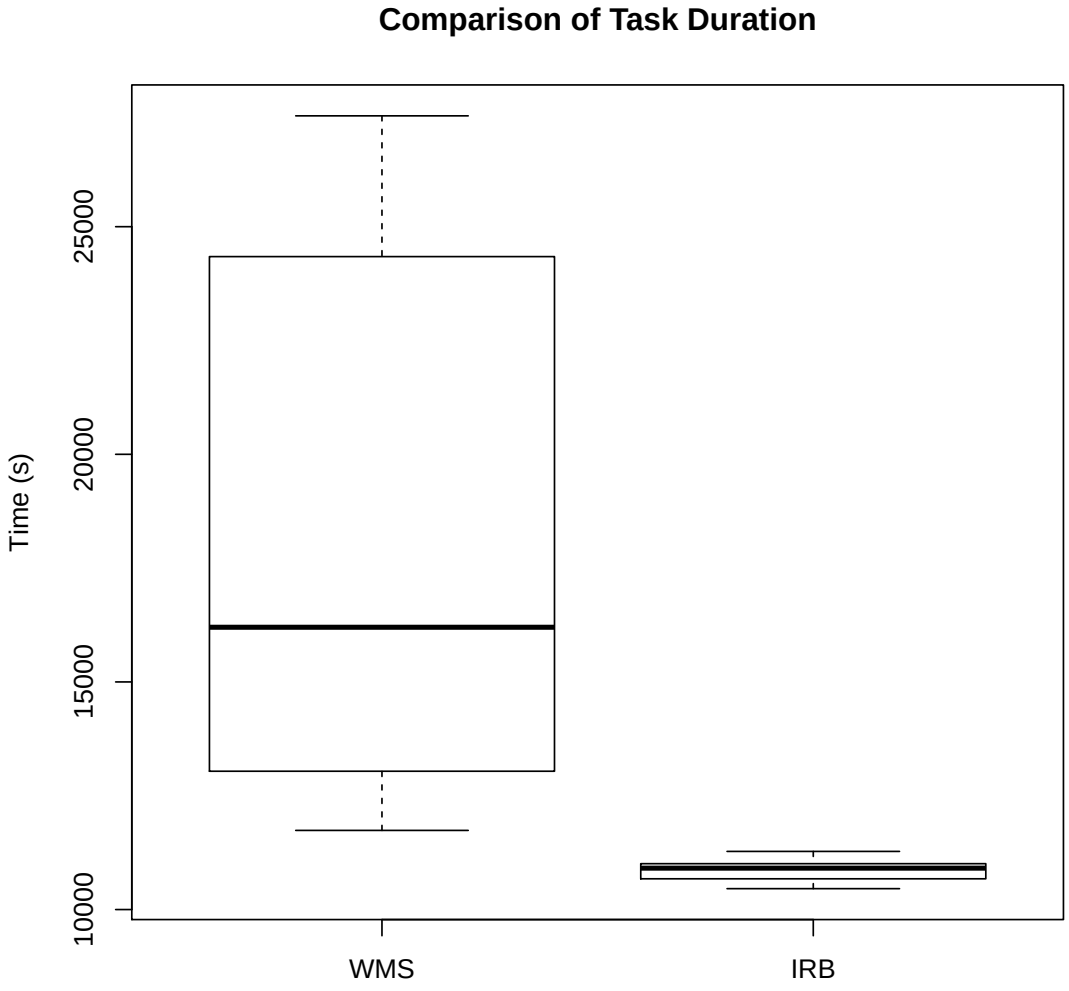

Figure 5.28 – Comparison of duration of jobs with downtime (type 2 tasks), illustrating the variation in mean task duration within the simulations. The lower and upper lines indicate the sample minima and maxima, the boxes indicate the extent of the second and third quartiles, and the solid lines denote the median values.

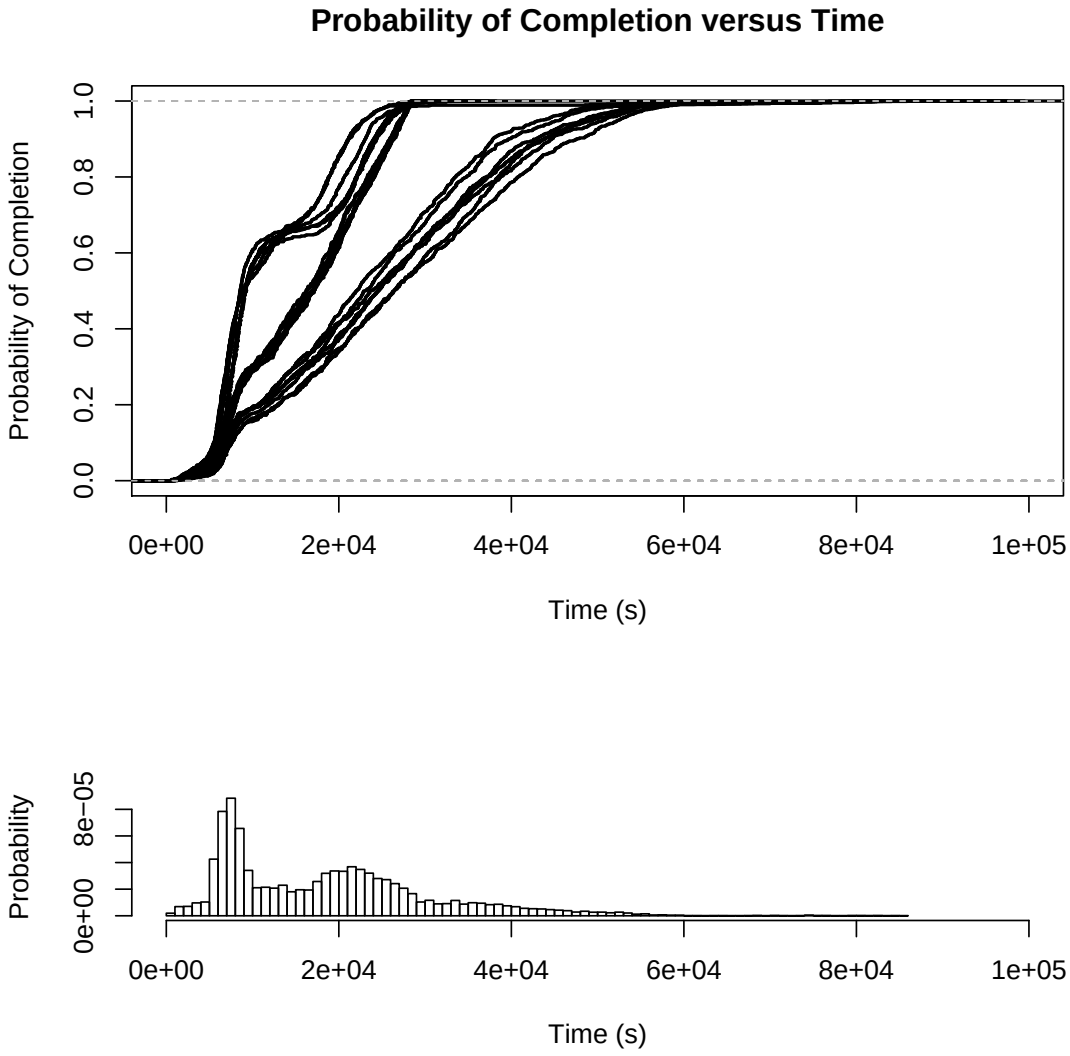

Figure 5.29 – Execution profile of jobs with downtime (WMS, type 2 tasks). Tasks which failed due to the effects of downtime are not shown.

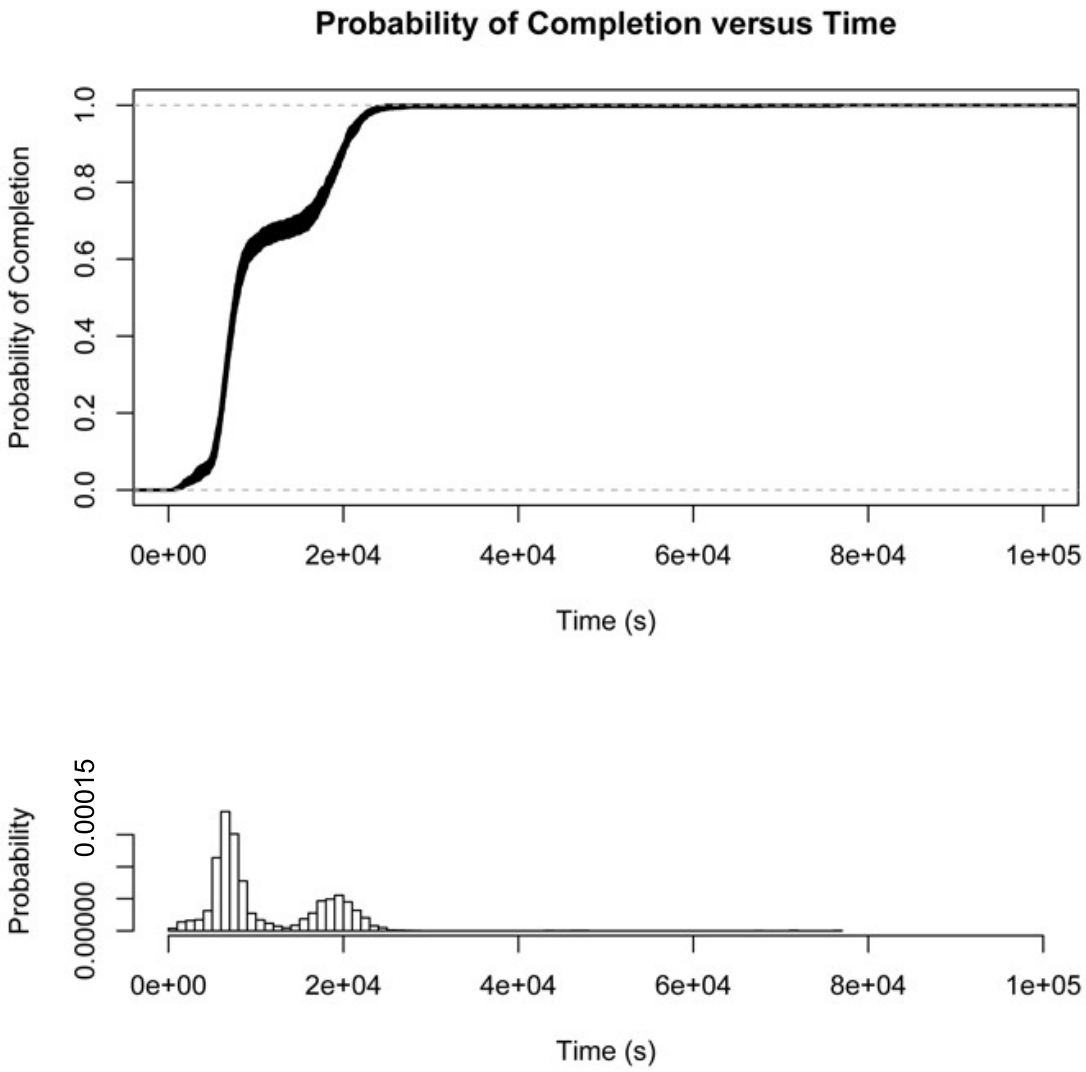

Figure 5.30 – Execution profile of jobs with downtime (IRB, type 2 tasks). Tasks which failed due to the effects of downtime are not shown.

5.3.6 Combined Requirements

The previous tests have examined the behaviour of the improved resource broker in the presence of various individual requirements; the only combination exhibited thus far has been to pair affinity with the other criteria. In order to investigate performance in a more realistic scenario, in which multiple requirements are present and interacting at once, a further series of tests was performed.

This example seeks to evoke a realistic context, where a user has a series of tasks to be run. As with previous examples, 1,000 tasks were used, and in this particular case they were type 2 tasks. The tasks were given a deadline three days hence, and so are representative of last-minute rush jobs for conference or deliverable deadlines. Each task required access to a licensed application, with licences being distributed as per *Table 5.17*. Two resources were listed as having scheduled downtime (*Table 5.18*), influenced by scheduled downtime on the resources' real-world counterparts intended to permit statutory electrical testing to be conducted.

Table 5.17 – Availability of licences on execution resources.

Resource	Licences
conan.elec.gla.ac.uk	500
miffy.elec.gla.ac.uk	500
ngs.leeds.ac.uk	No
ngs.ral.ac.uk	No
ngs.glasgow.ac.uk	48
ngs.manchester.ac.uk	200
ngs.lancaster.ac.uk	No
ngs.oxford.ac.uk	No
ngs.wmin.ac.uk	No
scotgrid.ac.uk	500

The success rate for a series of simulations using optimistic estimates of the duration of constituent tasks is shown in *Table 5.19* (extended summaries are provided in *Table E.20* and *Table E.21*). In this case, tasks managed by the current system are successful in a little over a quarter of instances. The scarcity of software licences accounts for the majority of task failures, with the adverse effects of downtime accounting for the remainder.

The improved resource broker apportions tasks among the three fastest, licensed resources, with the number of tasks allocated to each matching the number of licences

Table 5.18 – Downtime scheduled on execution resources (type 2 tasks).

Resource	Downtime
conan.elec.gla.ac.uk	+9.00 Hours – +25.00 Hours
miffy.elec.gla.ac.uk	+9.00 Hours – +25.00 Hours
ngs.glasgow.ac.uk	No
ngs.lancaster.ac.uk	No
ngs.leeds.ac.uk	No
ngs.manchester.ac.uk	No
ngs.oxford.ac.uk	No
ngs.ral.ac.uk	No
ngs.wmin.ac.uk	No
scotgrid.ac.uk	No

Table 5.19 – Job outcome: combined requirements (1,000 type 2 tasks).

	WMS	IRB
Completed	27.8%	99.0%
Failed due to missing licence	72.0%	0.0%
Missed deadline	0.0%	0.0%
Failed due to downtime	0.2%	1.0%

available (in the case of the two fastest resources), and with the remainder going to the third resource. The presence of scheduled downtime on two of the resources, coupled with the looming deadline, makes this the most suitable option. (Were the downtime to be any closer to the submission time, the practicality of using the two affected resources would be reduced given the typical length of a task. The only alternative would be to attempt to make use of the remaining, slower resources, despite the likelihood that the nominated deadline would be exceeded.)

Figure 5.32 illustrates the proportion of tasks which have completed as time progresses. As discussed previously in *Section 5.3.4 – Improved Resource Broker*, the nature of the distribution of type 2 tasks means that the occasional task of extreme duration makes it unlikely that all tasks will successfully complete when deadlines or downtime come into play.

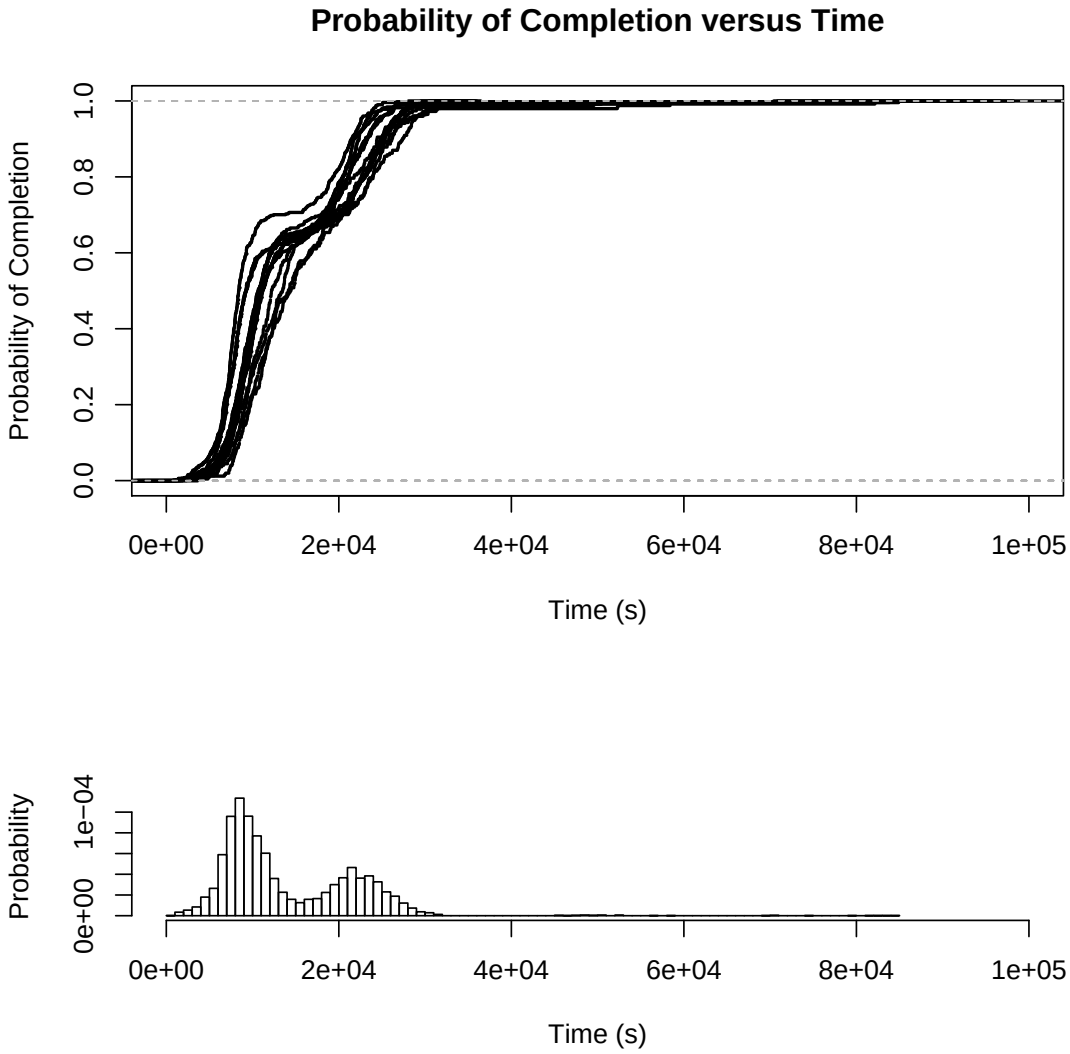

Figure 5.31 – Execution profile of jobs with combined requirements (WMS, 1,000 type 2 tasks). Tasks which failed due to licences not being available or due to the effects of downtime are not shown.

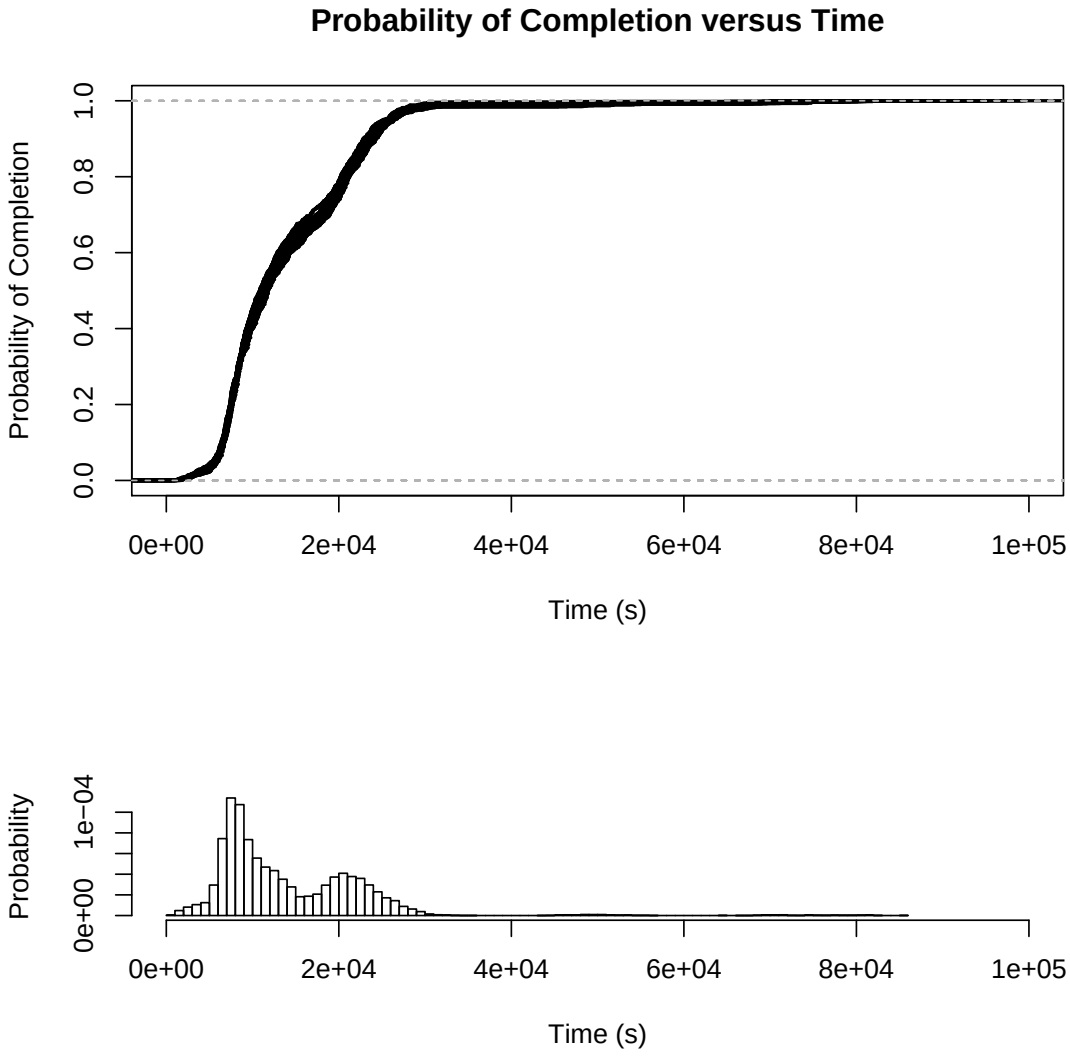

Figure 5.32 – Execution profile of jobs with combined requirements (IRB, 1,000 type 2 tasks).

If the job had been larger, the situation is more complicated yet again. Assuming that the user in this example wished to run a job comprising 2,000 tasks as opposed to 1,000 tasks (with all other requirements and conditions unchanged), the improved resource broker suggests an allocation of tasks which makes use of all licensed resources (see *Table 5.20*). In this instance, all available resources must be used in order to minimize the duration of tasks. Furthermore, the number of tasks allocated to particular resources (`conan.elec.gla.ac.uk` and `miffy.elec.gla.ac.uk`) is restricted due to the scheduled downtime.

Table 5.20 – Allocation of tasks to resources proposed by the improved resource broker (2,000 type 2 tasks). Where a second number is listed in parenthesis, the intention is that appropriate mechanisms must be used to prevent the number of simultaneous tasks exceeding this figure.

Resource	Number of Tasks
scotgrid.ac.uk	1,000 (500)
ngs.glasgow.ac.uk	96 (48)
miffy.elec.gla.ac.uk	4
conan.elec.gla.ac.uk	500
ngs.manchester.ac.uk	400 (200)

This example assumes a mechanism exists on the execution resources for the improved resource broker to place a restriction on the number of tasks which execute simultaneously, and that this is exposed to the grid infrastructure. Such functionality exists in many local resource managers, including later versions of Grid Engine [156], and the problem is discussed in a little more detail in *Appendix C – Integration with Local Job Managers*. If such a mechanism was not available, or was not suitably exposed to the resource brokering layer, a less sophisticated approach would have to be used. For example, the improved resource broker could submit tasks to resources in a series of appropriately sized batches, and only submit a new batch once the tasks in the previous batch had completed. This adds in effect an additional layer of queueing, increasing the complexity of the system and requiring the resource broker to undertake tasks which really should remain the purview of the local resource manager.

The success rate for this series of simulations is shown in *Table 5.21* (extended summaries are provided in *Table E.22* and *Table E.23*). The more complicated set of requirements conspires to work against the current system which is not equipped to handle them; in this case, only about one in ten tasks completes successfully. By comparison, the

Table 5.21 – Job outcome: combined requirements (2,000 type 2 tasks).

	WMS	IRB
Completed	9.9%	99.8%
Failed due to missing licence	90.0%	0.0%
Missed deadline	0.0%	0.0%
Failed due to downtime	0.1%	0.2%

overwhelming majority of tasks allocated to resources by the improved resource broker complete successfully. This behaviour can be seen in *Figure 5.33* and *Figure 5.34*.

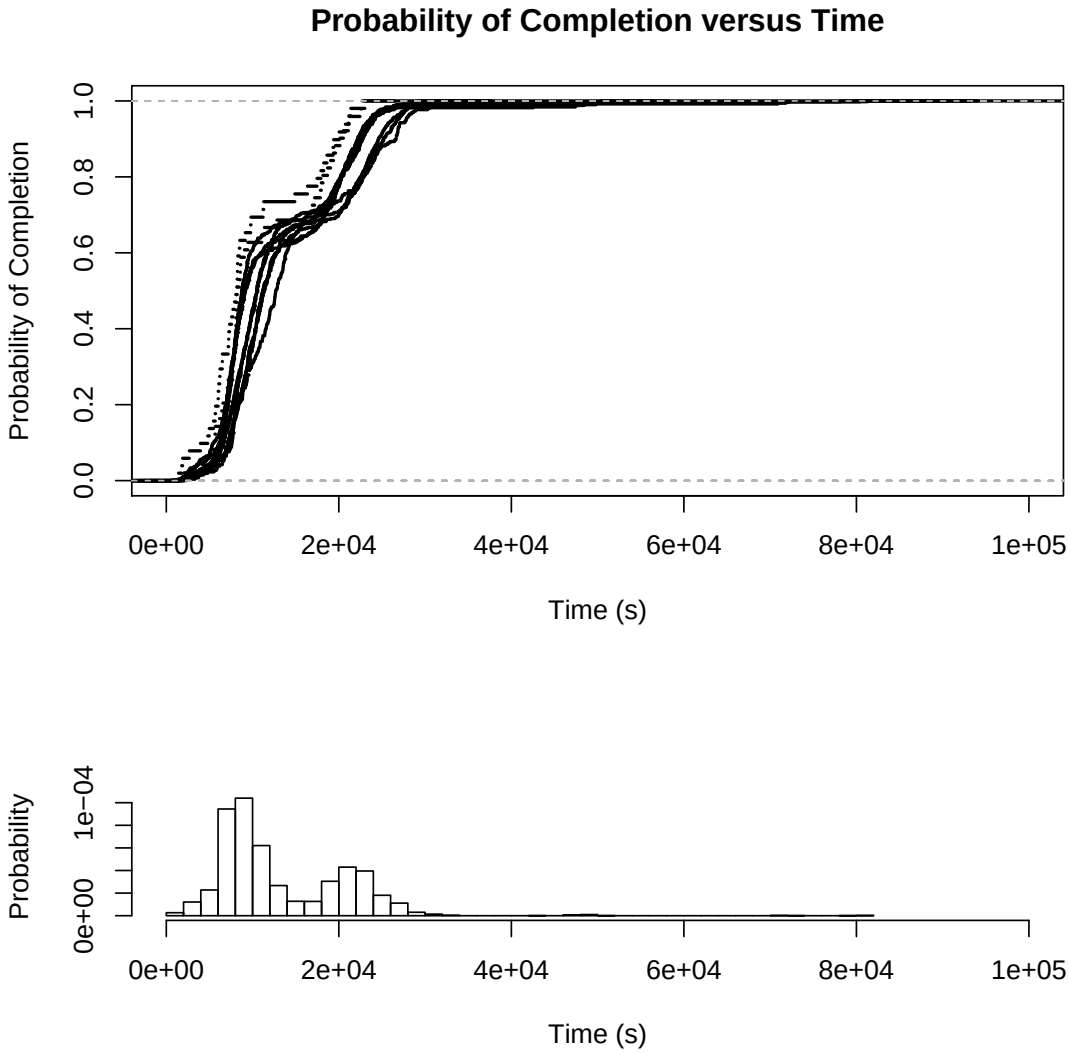

Figure 5.33 – Execution profile of jobs with combined requirements (WMS, 2,000 type 2 tasks). Tasks which failed due to licences not being available or due to the effects of downtime are not shown.

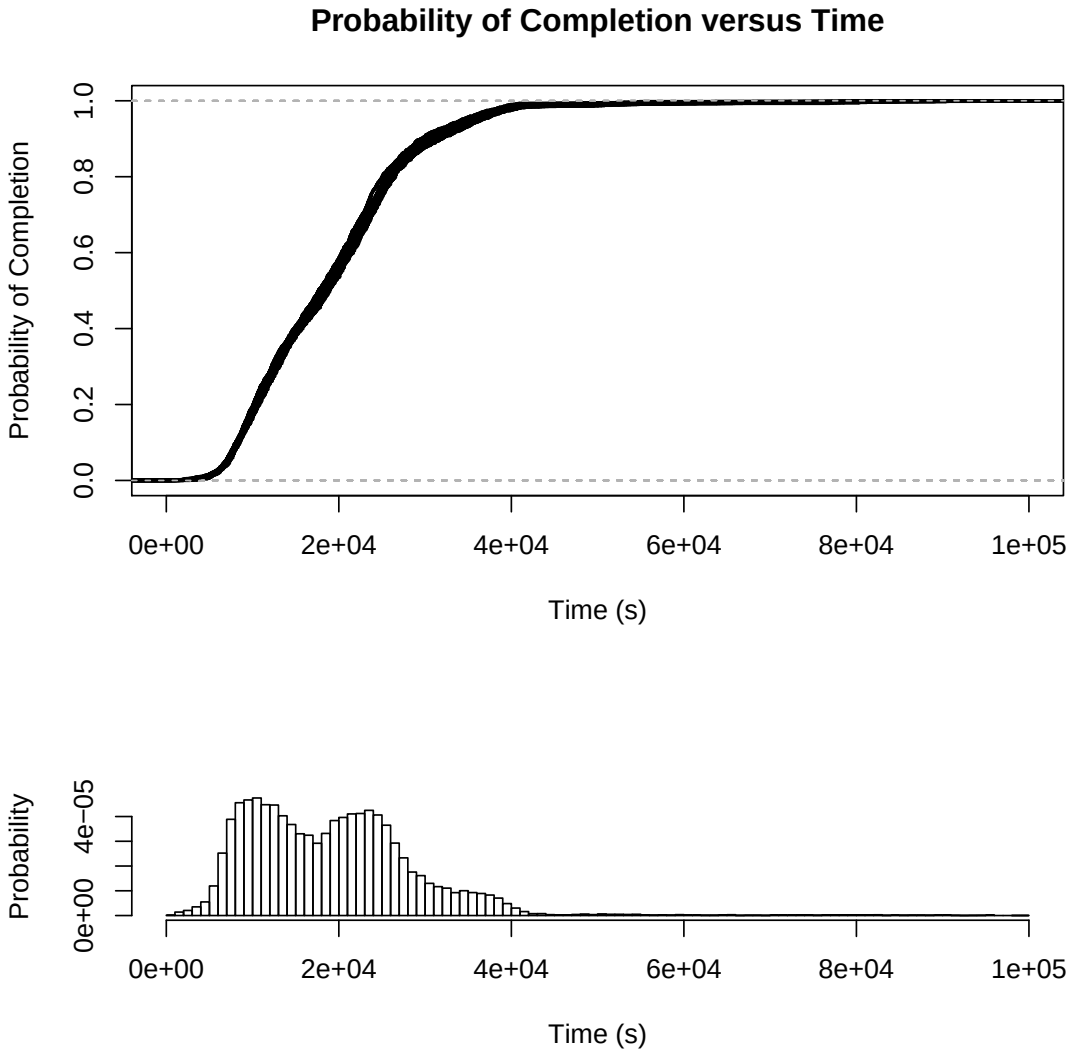

Figure 5.34 – Execution profile of jobs with combined requirements (IRB, 2,000 type 2 tasks).

Consider the case where a second user decides to submit 1,000 type 1 tasks shortly after the first user has submitted 1,000 type 2 tasks. These tasks have the same deadline as the first user's tasks. In this situation, the improved resource broker, conscious of the licences and resources used by the first set of tasks, proposes the allocation of tasks to resources described in *Table 5.22*. These type 1 tasks are of much shorter duration than the pre-existing type 2 tasks, and so the improved resource broker attempts to slot them into resources which have free licences; although these are the slower resources, the tasks will still be completed more quickly by using these resources than by waiting for the quicker resources to become available.

Table 5.22 – Allocation of tasks to resources proposed by the improved resource broker (1,000 type 1 tasks). Where a second number is listed in parenthesis, the intention is that appropriate mechanisms must be used to prevent the number of simultaneous tasks exceeding this figure.

Resource	Number of Tasks
miffy.elec.gla.ac.uk	592 (500)
ngs.manchester.ac.uk	400 (200)
conan.elec.gla.ac.uk	8 (4)

The outcome of tasks in this case is described in *Table 5.23* (full data may be found in *Table E.24* and *Table E.25*). Here, the matter of licences is the major factor affecting successful completion; the tasks are of much shorter—and much more consistent—duration, and consequently it is possible to ensure that they are unaffected by anticipated downtime. This is illustrated by *Figure 5.35* and *Figure 5.36*.

Table 5.23 – Job outcome: combined requirements (1,000 type 1 tasks).

	WMS	IRB
Completed	39.0%	100.0%
Failed due to missing licence	61.0%	0.0%
Missed deadline	0.0%	0.0%
Failed due to downtime	0.0%	0.0%

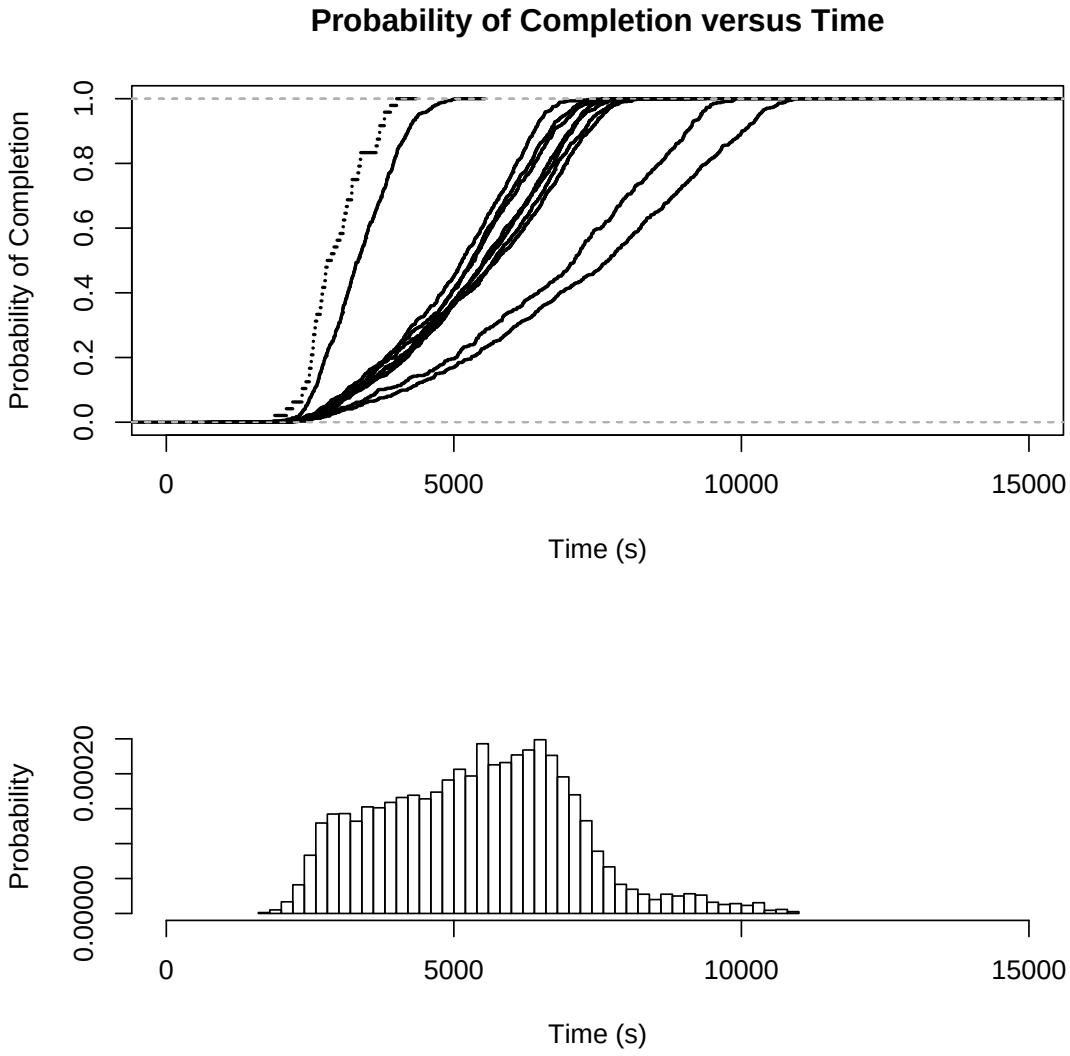

Figure 5.35 – Execution profile of jobs with combined requirements (WMS, 1,000 type 1 tasks). Tasks which failed due to licences not being available or due to the effects of downtime are not shown.

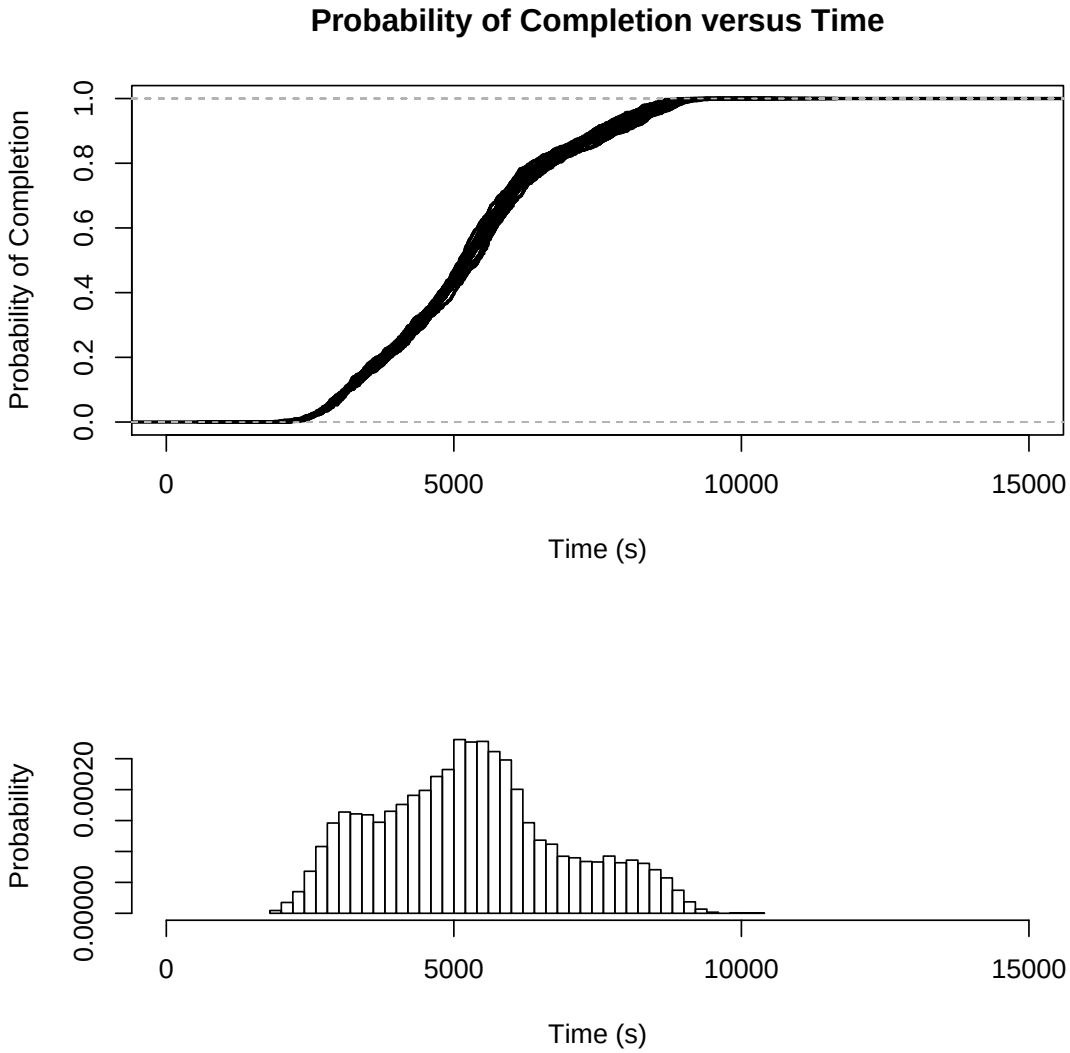

Figure 5.36 – Execution profile of jobs with combined requirements (IRB, 1,000 type 1 tasks).

5.3.7 AFS

The improved resource broker supports the inclusion of information about required AFS resources in a job submission document. Assume that the system exists in the state as described in *Table 5.24*, where all available resources have access to AFS and the necessary cell, but only those resources listed provide access to either a replica of the data (replicas in AFS are read-only), or host the cell itself.

Table 5.24 – Presence of AFS cells on execution resources. All other resources have access to the AFS cell, but do not replicate it locally.

Resource	Details
conan.elec.gla.ac.uk	Replica of `nesc.gla.ac.uk`
miffy.elec.gla.ac.uk	Replica of `nesc.gla.ac.uk`
scotgrid.ac.uk	Host of `nesc.gla.ac.uk`

In this scenario, the improved resource broker will favour `scotgrid.ac.uk`, as the file servers which host the cell are provided as part of this resource. It is therefore expected that this resource will offer the best read and write performance in relation to data held in that cell. The other resources listed (`conan.elec.gla.ac.uk` and `miffy.elec.gla.ac.uk`) provide read-only replicas of the cell, and will be used if, in the estimation of the improved resource broker, `scotgrid.ac.uk` was too busy to satisfactorily handle the new job. (As they are read-only, the replicas are of greater benefit when data is being read from AFS—such as, for example, if the applications to be used were installed in the AFS cell—than when it is being written to AFS.) If all three resources are busy, other resources which provide access to the cell will be used as would normally be the case.

A series of jobs was used to demonstrate the allocations produced by the improved resource broker; the allocations produced are described in *Table 5.25*. Type 1 tasks were used for this purpose, the resources were deemed to be under uniform load, and no additional requirements—such as licensing restrictions—were imposed.

With all other factors being equal, the allocations demonstrate a preference for the resource which hosts the AFS cell Thereafter, hosts which maintain replicas of the cell are used in preference to those which merely offer access. A complete evaluation of the performance of jobs with respect to their AFS requirements has not been conducted, as the test harness developed for evaluating the other principal aspects of the improved resource broker is not suitably developed for this particular use (see *Section 5.1.5 – Limitations of the Simulation*).

Table 5.25 – Task allocation: AFS requirements (type 1 tasks).

Number of Tasks	Allocation
1,000	scotgrid.ac.uk: 1,000
2,500	scotgrid.ac.uk: 2,112
	conan.elec.gla.ac.uk: 388
4,000	scotgrid.ac.uk: 2,112
	conan.elec.gla.ac.uk: 816
	miffy.elec.gla.ac.uk: 1,016
	ngs.leeds.ac.uk: 56

When access to more than one AFS cell is required in the course of execution of a job, the issues surrounding appropriate resource allocation become more complicated. If the needed AFS cells share common hosting and replication arrangements, there is little difference to the single-cell case; on the other hand, if access is required to AFS cells from multiple sources—for example, if access is required to cells provided by more than one institution—which may be hosted and replicated in different locations, an attempt is made to achieve a satisfactory compromise when allocating tasks to resources. In this case, additional information in the job description may be of benefit; for example, if it was known whether read-only or read-write access to a particular dataset was required, it would be possible to make a judgement as to the relative advantages of closeness to the origin of a cell as opposed to a replica. (In the case that merely read-only access is required, there is no disadvantage to be had by accessing an up-to-date replica as opposed to the original data source.) Some indication of the magnitude of the data to be accessed may also be useful.

A full exploration of the issues surrounding replica placement and data scheduling with regards to AFS is beyond the scope of this thesis. Extensions to GridSim exist which allow simulations to be modified to incorporate such aspects [195], so the evaluation methodology and test harness used earlier in this chapter could very well be adapted to explore this problem, given sufficient time and resources. It must be remembered that the performance of applications making extensive use of AFS is configuration-dependent; for example, small changes to cache sizes or differences in software versions can have significant effects on observed behaviour [67]. This means that considerable real-world study would be required in order to ensure that any simulation reproduced these effects authentically.

5.4 Restatement of Analysis

5.4.1 Affinity

The results in *Section 5.3.2 – Varying Affinity* demonstrate an improvement in task execution time and overall job throughput when use is made of the full range of available resources. In the evaluation, improvements of approximately thirty-five percent were observed with both short and long tasks. The use of the affinity attribute provides a clear way for a user to control the distribution of their job across the available computational resources. Similar functionality is provided in some commercial batch schedulers, but this is an important addition to the general job description format.

All tests were made with values of affinity of either 0.0 or 1.0. It might be interesting to investigate whether an intermediate setting (0.5, for instance) could be used to indicate that tasks may be divided among resources, but that the total number of resources used should be restricted. Possible use cases for this sort of functionality were presented earlier in this chapter.

5.4.2 Licensing

The results in *Section 5.3.3 – Licensed Applications* show an improvement in success rate when a naïve approach to the management of licensed applications is replaced with a more intelligent approach. The introduction of the licence element permits the licence requirements of a job to be specified easily when describing a job. Similarly, the personal licence element allows information about the availability of personal licences to be provided, which in turn allows personal licence models to be supported. (An example of such a model would be where the licence is attached to a particular set of data [51].) As demonstrated, this information can then be used when selecting resources for execution of the job. As discussed, the approach adopted is modular and flexible, permitting a range of adaptors to be developed to support and accumulate licence information from various sources. This would allow the improved resource broker to work with traditional licence managers, specialist managers designed for distributed and grid computing environments, and could also permit it to support bespoke solutions for keeping track of and controlling licence consumption on individual resources. Throughout, the emphasis has been on finding the means to support users' applications in their present forms, and to work alongside existing licensing technologies; the desired result is an improvement in the user experience which does not require radical alteration to the applications they wish to use, or to the manner in which they use them.

In the simulation, a significant improvement in terms of the percentage of tasks completing successfully was observed. Although it is not possible for a system such as the one described in this thesis—which works in conjunction with existing infrastructure, rather than replacing it—to always guarantee one hundred percent success, a definite benefit is still shown. Where the resource broker and batch scheduler are tightly integrated, appropriate steps can be taken to further increase this success rate (*Appendix C – Integration with Local Job Managers*).

5.4.3 Deadlines

The results in *Section 5.3.4 – Deadlines* demonstrate an improvement when resources are selected appropriately in order to manage jobs in the presence of a hard deadline, this deadline being specified by the user in the job description. The extent to which the actual durations of constituent tasks vary from the specified typical task duration will inevitably have an effect on the degree of success attainable, but even in the case when the variation is large a clear improvement can be seen; for example, simulation with type 2 tasks—which exhibit a distinctive, and awkward, double peak in the distribution of execution times—demonstrated an improvement in success rate from 87.8% to 98.8%. Given an appropriate affinity, multiple resources will be used, where available, to increase the likelihood that tasks will complete prior to the deadline.

The deadline estimation is complicated, not least as there are many factors which affect the calculation (such as differences between the estimated and actual duration of tasks, and variations in queue times). However, this functionality can be important when jobs are required to be run with a degree of urgency, as not only will resources be selected in such a way as to provide the best opportunity of meeting the deadline, but it will often present an opportunity to alert the user to the fact that it will not be possible to meet some deadline prior to submission of a package of work, which should provide time for alternative options to be sought and considered. Such options could include recourse to a commercial resource offering greater capacity, but with associated financial implications.

The concept of deadline is very important to many economical models, a facet of distributed computing not explored by this work. In such a model, it is often the case that a trade-off can be made between cost and deadline (i.e. a job may be completed more quickly if a greater quantity of resources is used simultaneously, but this will incur increased usage charges) [35, 36]. This functionality would therefore be of benefit were it desirable to extend the system to have some understanding of resource cost models.

5.4.4 Downtime

The results in *Section 5.3.5 – Downtime* demonstrate that predictable downtime can be taken into consideration at the resource selection stage, and in many cases its effects can be successfully mitigated. The techniques used to work around downtime are essentially similar to those used for deadlines, but with the exception that the deadline can vary from one resource to another, and further that the resource may become available again at some point in the future. The degree of success depends on how cautious the estimates of anticipated task duration are: the more conservative the estimates, the greater the overall success rate will be, although there is the potential for resources to be excluded from consideration despite the fact they could offer useful computational capacity.

Of course, downtime which is unplanned—for example, that resulting from hardware failure or disruption to power supplies and cooling—cannot be predicted, and therefore cannot be managed by this system, or indeed by any similar system. That is not to say that the effects of unexpected downtime cannot be mitigated at all; systems such as SPHINX [110] attempt to recover from such downtime by rescheduling affected jobs on other resources. However, when working with large resources there will always be the need to perform maintenance or other administrative tasks necessitating the shutdown of the whole or part of the resource, and the ability to incorporate such knowledge into the resource brokering process—and thereby avoid the need for the user to restrict jobs accordingly, or indeed to attempt recovery following downtime—is beneficial.

5.4.5 Combined Requirements

The results in *Section 5.3.6 – Combined Requirements* show that the improvements illustrated by the earlier tests become more impressive when a combination of requirements or conditions, akin to those which might reasonably govern the behaviour of tasks in a realistic scenario, are placed upon a series of tasks. When presented with multiple requirements—licensing issues, deadlines and scheduled downtime—it was shown that the improved resource broker was able to secure a high degree of success even in the sort of challenging circumstances which originally inspired much of this work.

5.4.6 AFS Requirements

Although a full performance evaluation has not been conducted, the results in *Section 5.3.7 – AFS* demonstrate the awareness of AFS requirements exhibited by the improved resource broker. These requirements are specified by the user in the job description prior to job submission. The extensions to JSDL used to describe these requirements

are sufficient to capture basic information, such as the cells needed and whether the use of an encrypted connection is mandatory, which is enough in the first instance to satisfy the needs of projects which wish to make use of AFS [182].

It is likely that further extensions to JSDL, permitting additional details about data access operations to be included in a submission document, would be useful, particularly when there is the need to access data from more than one cell. Such extensions would provide the ability to specify details such as the amount of data to be transferred, and whether read-only or read-write access is required. It has been shown in the past that AFS can be extremely useful in distributed and grid computing projects [113, 182], and so any improvement in support for this technology can only be of benefit.

5.5 Summary

This chapter introduced GridSim, a tool used to simulate the operation of distributed computing systems, and described how it was adapted to perform the evaluation for this work. It described the background to the types of job represented in the evaluation, demonstrating that these are drawn from real-world examples. It then discussed the various tests that were performed, before describing each test and the accompanying results in detail, as well as providing some information about the applicability of these results to real-world workflows. Finally, the analyses of results presented during the commentary to the evaluation were summarized.

Chapter 6
Conclusions

6.1 Principal Conclusions

This thesis has described the development of a tool which allocates jobs to resources according to an assessment of suitability which takes into account a wider range of information than other comparable tools, such as the Workload Management System (WMS) [136]. This information is drawn from documents which accord to the JSDL specification, including both accepted and bespoke extensions to this specification. The intention is to improve the end-user experience for people who wish to run large ensembles of jobs, particularly in cases where some knowledge of the typical behaviour of such jobs is possessed beforehand, but without requiring significant additional effort on the part of the user, or modification to existing computational infrastructure. For the purposes of evaluation, examples have been drawn from the field of electronic engineering, and specifically from applications featured in the project *Meeting the Design Challenges of Nano-CMOS Electronics*. There are, however, many similar examples of relevant high-performance and high-throughput computing problems, in fields ranging from image retrieval [203] to transport modelling [140].

The evaluation has shown how some consideration of this additional information can lead to significant improvements in terms of successful completion rates and execution times, particularly when submitting arrays of similar tasks (a common situation, especially in the field of high-throughput computing where array and parameter-sweep jobs are prevalent). In particular, examples illustrating task affinity, provision of licences, and awareness of deadlines and impending system downtime have been explored. In each case, improvements in terms of task completion and throughput—and consequently on the whole user experience—have been shown. The use of affinity settings to permit easy division of tasks among resources demonstrated a thirty-five percent reduction in mean task duration, a significant improvement. The introduction of licence specifications in

the job description permit these to be taken into consideration during allocation, and as a consequence raising the success rate to close to one hundred percent. The addition of support for deadline and downtime criteria ensured similar increases in success rate. Furthermore, the system was shown to be capable of handling situations where multiple requirements were combined, and in these cases the potential improvements were shown to be even greater. Lastly, it was shown how support for AFS, a distributed file-system, could be incorporated, and some discussion of its manner of operation was provided.

6.2 Thesis Statement Revisited

For ease of reference, the thesis statement presented at the beginning of this work is restated here:

> The author asserts that the current generation of grid schedulers and resource managers fail to consider fully the diverse range of requirements that have some bearing on the success of any submitted job. These include the requirements of other, competing, uses of a resource, the location of required data, the authorization-related rights of a user, and the licences of the applications and data involved.
>
> It is feasible to express these requirements prior to job submission, and to extract information from the execution resource necessary to determine the likelihood that such requirements can be satisfied or otherwise managed, in order that the allocation of jobs to resources—and, ultimately, the experience of users—may be improved.
>
> This research investigates and proposes methods by which the above can be achieved, and seeks to prove the effectiveness of these proposals by implementing a modified resource brokering system which takes account of this information. This has been incorporated within a test infrastructure, in order that effective comparisons can be made between existing and proposed techniques.

After introducing the field and explaining the motivation for this work (*Chapter 1 – Introduction*), and then surveying the diverse range of technology that is involved in distributed and grid computing (*Chapter 2 – Background and State-of-the-Art*), a range

of potential requirements that are not given due consideration in current-generation systems was explored (*Chapter 3 – Proposed Requirement Framework*). These requirements were presented alongside a series of short case studies describing their applicability to real-world scenarios, and additional information necessary to evaluate the suitability of execution resources in the light of such requirements was identified.

The design for an improved resource broker was then presented (*Chapter 4 – Design of an Improved Resource Broker*). This design evolved from the requirements specified previously and, where it had been identified that additional information would need to be gathered during the job submission process, extensions to an existing, standard job description format were proposed. The design was further developed, providing details of particular components and their interrelationships, and emphasizing the aim that the resultant system should embody the flexibility needed to support a domain which contains a myriad different technologies, and which is constantly changing and developing.

In order to test the assertion that consideration of these additional requirements would lead to an improvement in the allocation of jobs to resources, an evaluation was designed and carried out (*Chapter 5 – Evaluation*). This process began with a discussion of the relative merits of simulation as an evaluation technique, before the implementation of a simulation test harness was described in detail. The selection of the applications to be used during the evaluation was justified, supported by information from a project in the nano-scale semiconductor simulation domain, which made heavy use of these applications in a grid computing context. A number of tests were planned and undertaken in order to evaluate the influence that the additional job requirements have on the proposed allocation of jobs to resources. It was shown, by means of simulation, that there were significant improvements in the success rate and performance of jobs when compared to an existing grid resource broker.

Returning to the original thesis statement, it has been shown that there are indeed additional constraints and requirements governing the jobs typical of real academic and industrial research projects, and that these can be gathered during the job submission process. A method for specifying these additional requirements has been provided, which builds upon and enhances the existing JSDL standard. Furthermore, it has been demonstrated that it is possible to give due consideration to these requirements when allocating jobs to resources, and that if this is done there will be a beneficial effect on the overall throughput of these jobs on these resources. This in turn will have a positive effect on users' experiences of grid computing as a whole. The flexible, modular approach adopted

by the improved resource broker helps to ensure that it is capable of supporting and interacting with a wide range of current and future technologies which make up the grid and distributed computing field.

6.3 Future Work

There are several areas touched upon by this thesis which would merit further investigation. One such example is the affinity setting, which describes whether or not different tasks from one job may be divided among more than one resource. This is defined as a floating point value, but treated in this work as a boolean value. It would be interesting to establish whether values greater than 0.0 but less than 1.0 could be used effectively to control the extent to which tasks may be split up. For example, a value towards the higher end of the scale would suggest that a division of tasks is permissable, but that the number of resources used should be minimized. Use cases for functionality of this sort were presented earlier, but the idea was not further developed by this thesis.

This thesis has examined additional factors which, when taken into consideration by resource brokers, can offer improvements to the speed or throughput of jobs. It has purposefully refrained from examining in any great detail the process by which these factors are evaluated. As part of this work, a simple 'constraint optimizer' has been developed, but it is undoubtedly the case that this is an area in which knowledge drawn from the field of constraint programming, a significant area of computer science research, could be beneficial. For example, CHOCO [114] is a constraint programming tool providing a library of functions implemented in Java, and would be an ideal candidate for integration into the improved brokering tool.

There is also some work to be done in order to make the tools as they exist at present suitable for more general use (that is, to transform them from research code into production-quality applications). The main deficiency is that there is no integration of the code which evaluates the job submission documents and suggests an allocation profile, and the applications which perform the actual submission of the jobs. Job submission is therefore a two-step process, with the output from the first stage being manually fed into the second. It should be a relatively uncomplicated task to integrate these two steps in order that a user could submit jobs in one simple action; indeed, applications which perform similar operations were developed as part of the related project, *Meeting the Design Challenges of Nano-CMOS Electronics* [209]. Furthermore, by taking advantage of a tool such as

SAGA [89], it should be possible to support a wide range of grid middleware with minimal effort. Other enhancements might include a user-friendly interface through which JSDL descriptions may be constructed, saving the user the chore of hand-crafting these admittedly rather arcane documents.

Finally, it might be interesting to expand the evaluation test framework to model certain cloud resources, such as Amazon Elastic Compute Cloud (EC2) services[1]. Many of the issues affecting grid computing are also present in cloud computing, and so the suggestions made in this thesis would still be of benefit in this emerging domain. Such additions to the test framework may necessitate improving its modelling of network latency and data transfer performance, and this in turn would permit a more thorough analysis of the behaviour of jobs dependent on AFS to be made.

[1] http://aws.amazon.com/ec2

Appendix A
JSDL Extension Schema

Where differences exist between the definition below and the description provided in the main text, the contents of this appendix must be considered normative.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.nesc.gla.ac.uk/jsdl/2010/02/gps"
            xmlns:jsdl-gps=
                    "http://www.nesc.gla.ac.uk/jsdl/2010/02/gps"
            xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
            targetNamespace=
                    "http://www.nesc.gla.ac.uk/jsdl/2010/02/gps"
            elementFormDefault="qualified" version="1">

    <xsd:import
          namespace="http://schemas.ogf.org/jsdl/2007/04/sweep"
          schemaLocation="./sweep.xsd"/>
    <xsd:import
          namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl"
          schemaLocation="../jsdl/jsdl.2005_11.xsd"/>

    <xsd:complexType name="AFS_Type">
        <xsd:sequence>
            <xsd:element name="cell" type="xsd:string"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="encryption"
                         minOccurs="0" maxOccurs="1">
                <xsd:attribute name="enabled"
```

```
                                type="xsd:boolean"/>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Licence_Type">
        <xsd:sequence>
            <xsd:element ref="jsdl:ApplicationName"
                         minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="jsdl:ApplicationVersion"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="number" type="xsd:integer"
                         minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Personal_Licence_Type">
        <xsd:sequence>
            <xsd:element ref="jsdl:ApplicationName"
                         minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="jsdl:ApplicationVersion"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="number" type="xsd:integer"
                         minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="Affinity_Type">
        <xsd:restriction base="float">
            <xsd:minInclusive value="0.0"/>
            <xsd:maxInclusive value="1.0"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- ==================== -->
    <!--<xsd:element name="AFS" type="AFS_Type"/>-->
    <xsd:element name="Licence" type="Licence_Type"/>
```

```
    <xsd:element name="PersonalLicence"
                 type="Personal_Licence_Type"/>
    <xsd:element name="Affinity" type="Affinity_Type"/>
    <xsd:element name="Deadline" type="xsd:dateTime"/>

</xsd:schema>
```

Appendix B
Program Design

The software developed as part of this thesis has been divided into packages of related functionality. The overall package structure is illustrated in *Figure B.1*. Core class relationships are shown in a series of class diagrams: the main program (*Figure B.2*), the licence management components (*Figure B.3*), schedule-prediction components (*Figure B.4*) and one of the evaluators (*Figure B.5*). Not all classes have been illustrated—those from the standard library and from third-party sources have been omitted entirely—and a class may be represented in more than diagram where it provides functionality to different parts of the program.

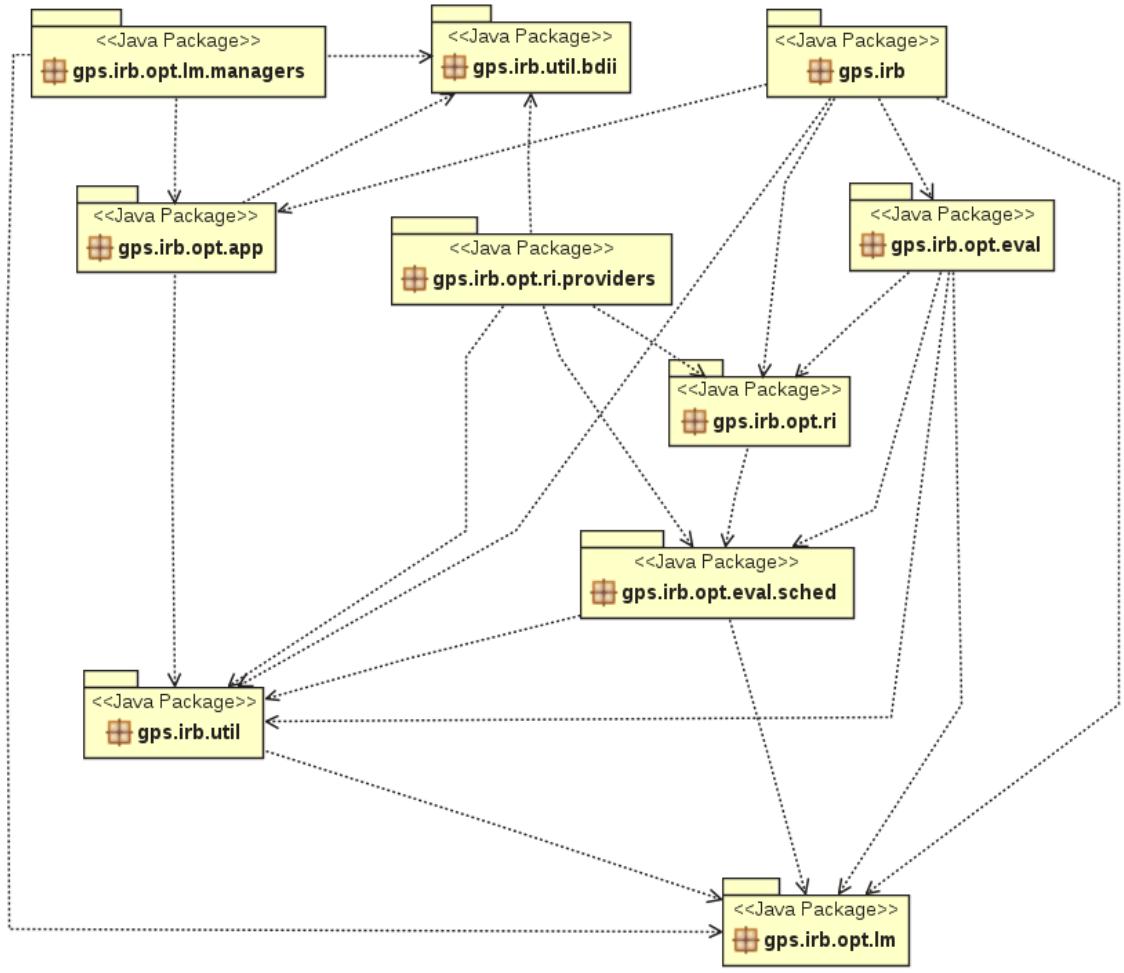

Figure B.1 – Package diagram. Packages forming part of the standard library and those provided by third parties are not included.

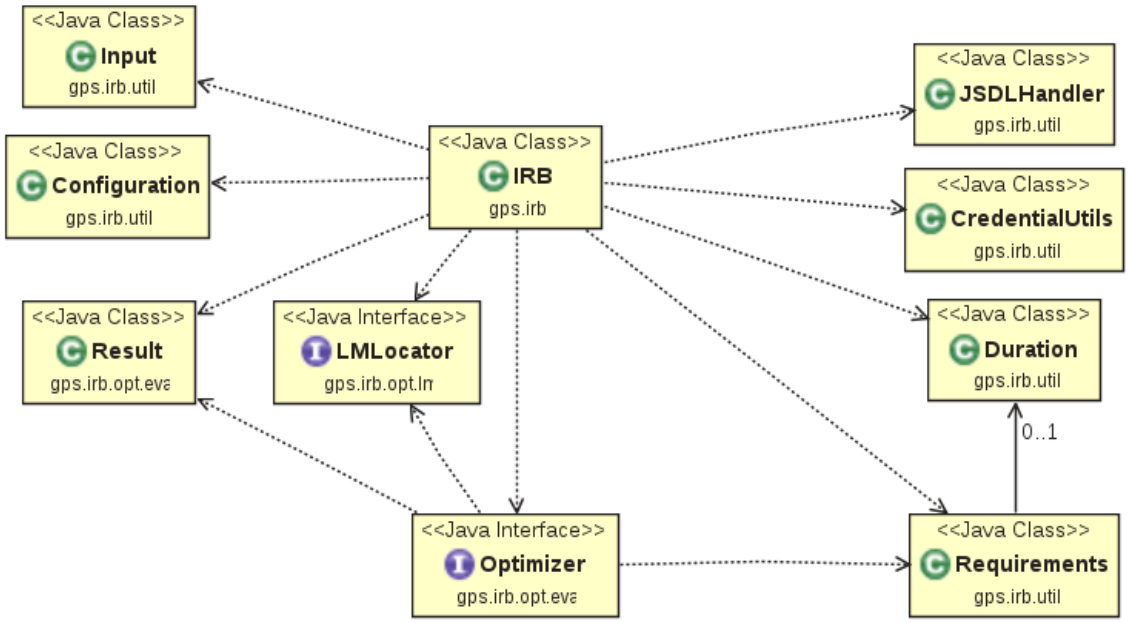

Figure B.2 – Class diagram: main program. Classes forming part of the standard library and those provided by third parties are not included.

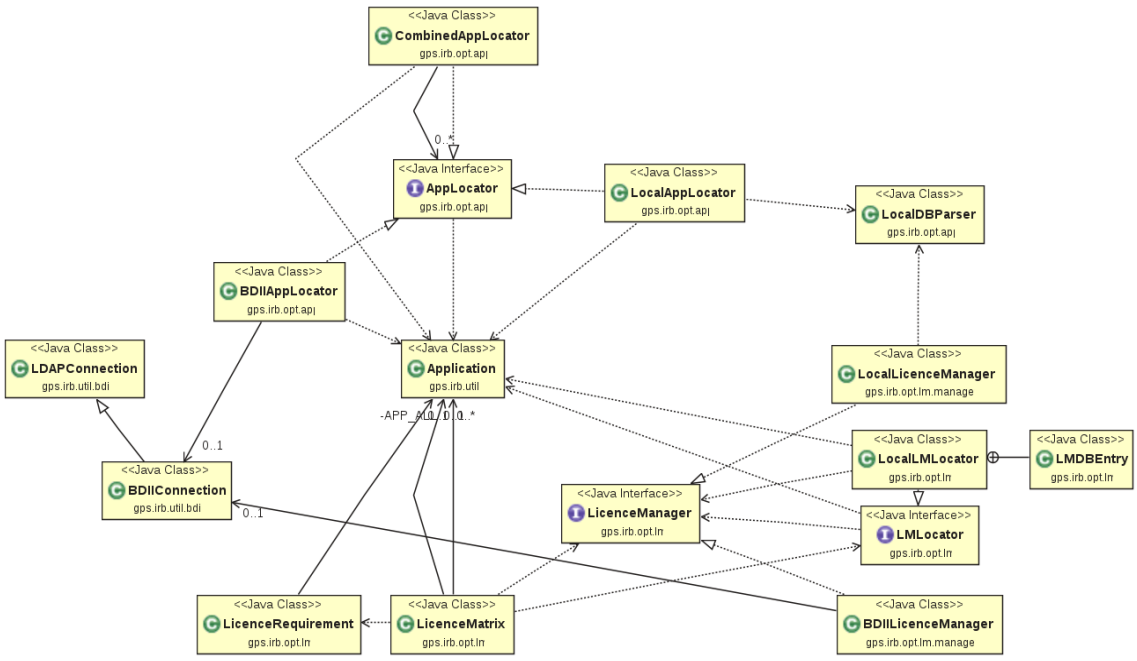

Figure B.3 – Class diagram: licence management components. Classes forming part of the standard library and those provided by third parties are not included.

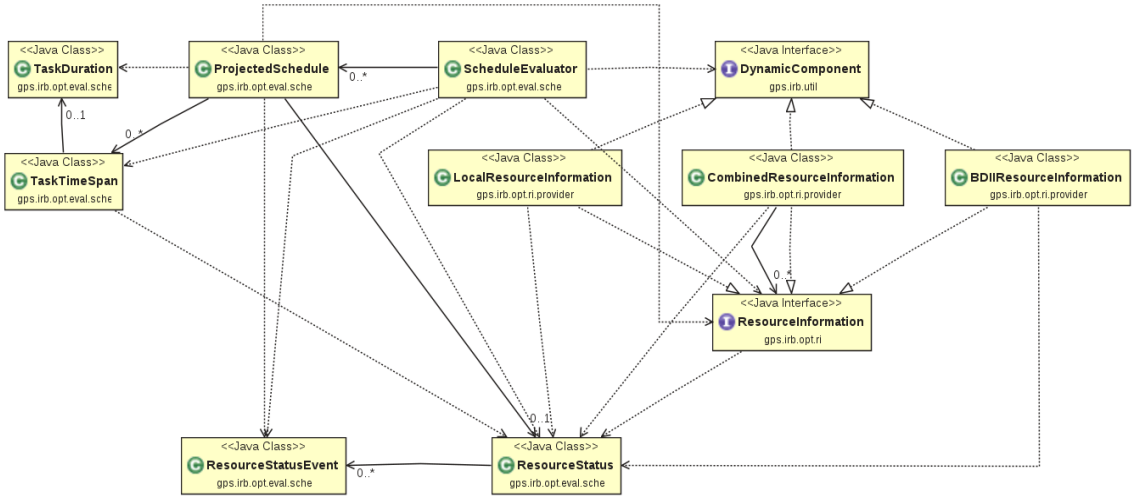

Figure B.4 – Class diagram: schedule-prediction components. Classes forming part of the standard library and those provided by third parties are not included.

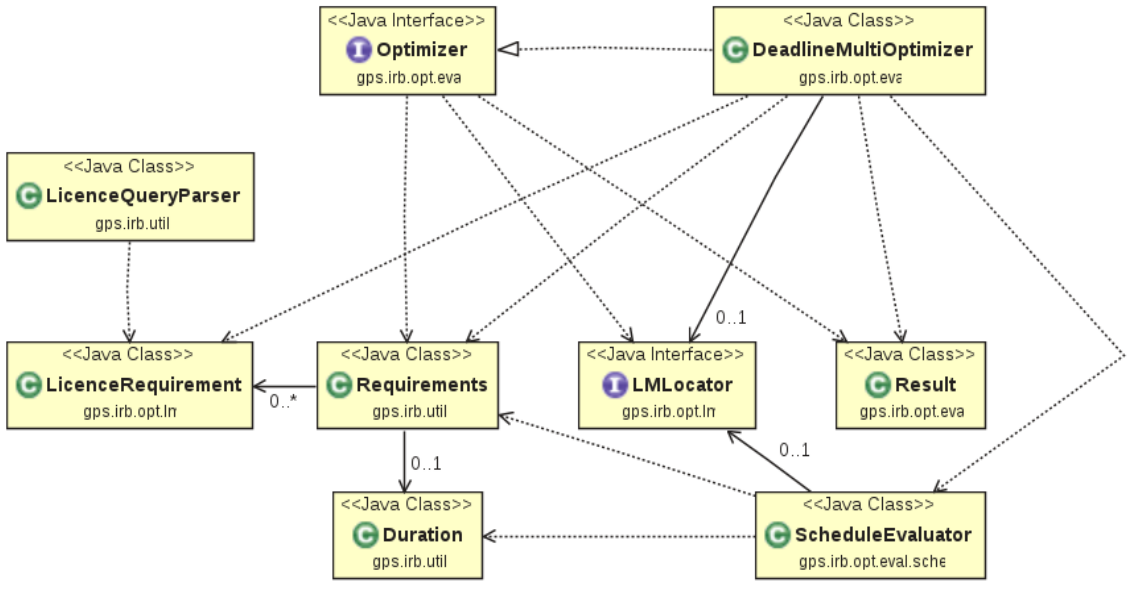

Figure B.5 – Class diagram: one of the evaluators. Classes forming part of the standard library and those provided by third parties are not included.

Appendix C
Integration with Local Job Managers

The extent to which functionality such as parameter sweeps is supported depends on the capabilities of the applications used by resources to schedule jobs locally (see *Section 2.6.7 – Local Job Scheduling*). Popular grid software such as the Globus Toolkit provides the mechanisms required to integrate with local schedulers of this sort. However, supporting the full range of desirable features may require significant customization, or the need to resort to the use of bespoke adapters. Furthermore, there will always be particular features in one local resource manager which are not provided by another, making the task of developing higher-level resource brokers which seamlessly support a range of local resource managers rather complicated.

This appendix briefly describes the means by which requirements in job descriptions can be translated to local submission documents for one particular family of local job managers, those in the family of tools which descended from Grid Engine (for a very brief history, see *Section 2.6.7 – Local Job Scheduling*). This is not intended to provide an introduction to Grid Engine (instead, refer to the user guide [156]), nor is it intended to be a comprehensive design; rather it is intended to hopefully clarify some of the particular difficulties with integrating generic grid tools with specific local environments. This is based upon work done for the research project *Meeting the Design Challenges of Nano-CMOS Electronics*, for which a bespoke service converting JSDL job descriptions to Grid Engine submission documents was produced.

A Grid Engine job description takes the form of an annotated shell script. This script includes the commands which must be run to execute whatever applications the user desires, decorated with some directives which tell Grid Engine how to handle the job

(controlling things such as what the job should be called and how its standard output streams should be managed).

Most of the JSDL elements from the JSDL POSIX application schema map neatly to Grid Engine equivalents. For example, the `<Executable>` and `<Argument>` elements can be used to generate the portion of the shell script which runs the application, while the `<Output>` and `<Error>` elements can be used to populate Grid Engine's `-o` and `-e` arguments respectively.

Decisions such as the selection of an appropriate queue in which to run an application should a choice of queues exist, are not typically supported by JSDL, as these are local issues specific to a particular resource, and are not something which a remote user (who may have little or no knowledge of how the resource is configured) should have to concern themselves with. Appropriate queues and, where necessary, parallel environments should therefore be selected automatically when translating JSDL specifications into Grid Engine submission documents.

Grid Engine supports parameter sweeps where the parameters are integers forming an arithmetic sequence. Simple sweeps defined using the JSDL `<LoopInteger>` element from the parameter sweep extension [61] can be supported by directly mapping the `start`, `end` and `step` attributes to Grid Engine's `-t start-end:step` argument.

More complex parameter sweeps (using the likes of the `<LoopDouble>` and `<Values>` sweep functions, or by introducing the `<Except>` element to an integer sweep) present an additional complication, as these are not supported directly by Grid Engine. Instead, the approach adopted is to submit a job with an integer parameter sweep of equivalent size, and use this as an index into an array which can be accessed when the job runs in order to obtain the actual parameters to be used. This array can be included in the submission file itself, or written to a separate file generated automatically by the submission service.

Licensing presents further issues which need to be considered. Local resources may keep track of licences independently of the grid infrastructure, in an attempt to prevent more jobs running than there are available licences; in Grid Engine, this could take the form of a consumable resource which licensed jobs deplete when they begin to run, and replenish upon completion. Jobs requesting this resource are held in the queue until a sufficient quantity becomes free. This could be used in conjunction with resource reservation to prevent starvation of jobs requiring a greater quantity of resources than is the norm. A

bespoke solution such as this naturally necessitates some degree of manual integration with a generic grid scheduler, so that it is aware what resources need to be requested in order to ensure that licences will be made available.

An alternative, simpler (albeit less fool-proof) approach to the problem may be to allow the grid scheduler to restrict the number of instances of a job which run simultaneously (for example, to permit the grid scheduler to submit an array job comprising five hundred tasks, but to make the proviso that no more than fifty tasks should run concurrently). Again, this requires appropriate functionality in the underlying local job scheduler; the `-tc` argument, which provides just this functionality, has been available in Grid Engine since version 6.2 update 4.

Appendix D
Example Output

```
Optimizer: gps.conopt.opt.eval.TotalNoSingleOptimizer
[Equal]
+ [scotgrid.ac.uk: 1000] Score: 3.28084E-5
+ [ngs.lancaster.ac.uk: 1000] Score: 3.28084E-5
+ [miffy.elec.gla.ac.uk: 1000] Score: 3.28084E-5
+ [conan.elec.gla.ac.uk: 1000] Score: 1.64042E-5
+ [ngs.wmin.ac.uk: 1000] Score: 8.2021E-6
+ [ngs.ral.ac.uk: 1000] Score: 8.2021E-6
+ [ngs.oxford.ac.uk: 1000] Score: 8.2021E-6
+ [ngs.manchester.ac.uk: 1000] Score: 8.2021E-6
+ [ngs.leeds.ac.uk: 1000] Score: 8.2021E-6
+ [ngs.glasgow.ac.uk: 1000] Score: 8.2021E-6

Optimizer: gps.conopt.opt.eval.TotalNoSingleOptimizer
[BestFit]
+ [ngs.lancaster.ac.uk: 1000] Score: 2.3736265
+ [scotgrid.ac.uk: 1000] Score: 1.8992807
+ [conan.elec.gla.ac.uk: 1000] Score: 1.2254902
+ [miffy.elec.gla.ac.uk: 1000] Score: 0.984252
+ [ngs.wmin.ac.uk: 1000] Score: -0.34408602
+ [ngs.oxford.ac.uk: 1000] Score: -0.34408602
+ [ngs.manchester.ac.uk: 1000] Score: -0.34408602
+ [ngs.leeds.ac.uk: 1000] Score: -0.34408602
+ [ngs.glasgow.ac.uk: 1000] Score: -0.37362638
+ [ngs.ral.ac.uk: 1000] Score: -0.40449437

Optimizer: gps.conopt.opt.eval.TotalNoSingleOptimizer
```

```
[WorstFit]
+ [scotgrid.ac.uk: 1000] Score: 999.0
+ [ngs.lancaster.ac.uk: 1000] Score: 999.0
+ [ngs.wmin.ac.uk: 1000] Score: 231.0
+ [ngs.oxford.ac.uk: 1000] Score: 231.0
+ [ngs.manchester.ac.uk: 1000] Score: 231.0
+ [ngs.leeds.ac.uk: 1000] Score: 231.0
+ [ngs.glasgow.ac.uk: 1000] Score: 183.0
+ [ngs.ral.ac.uk: 1000] Score: 135.0
+ [miffy.elec.gla.ac.uk: 1000] Score: 0.015748031
+ [conan.elec.gla.ac.uk: 1000] Score: -0.2254902

Optimizer: gps.conopt.opt.eval.TotalNoMultiOptimizer
[Equal, AcceptPartial]
+ [miffy.elec.gla.ac.uk: 500, conan.elec.gla.ac.uk: 500]
  Score: 1.0

Optimizer: gps.conopt.opt.eval.TotalNoMultiOptimizer
[BestFit]
+ [miffy.elec.gla.ac.uk: 1000] Score: 1.0

Optimizer: gps.conopt.opt.eval.TotalNoMultiOptimizer
[WorstFit]
+ [miffy.elec.gla.ac.uk: 1000] Score: 1.0

Optimizer: gps.conopt.opt.eval.DeadlineMultiOptimizer
[Equal, AcceptPartial]
+ [ngs.glasgow.ac.uk: 184, conan.elec.gla.ac.uk: 816] Score: 1.0
```

Appendix E
Evaluation Results

This appendix summarizes some of the principal results referred to in the evaluation presented earlier in this thesis (see *Chapter 5 – Evaluation*). Within the tables of results, the following definitions apply:

S: Percentage of tasks which completed successfully.

L: Percentage of tasks which failed due to lacking required licences.

DL: Percentage of tasks which did not complete before their deadline.

DT: Percentage of tasks which did not complete prior to occurrence of scheduled downtime.

Mean Execution Time: Arithmetic mean of task wallclock time, in seconds (i.e. the amount of time which passes between a task beginning to execute, and its termination).

Mean Duration: Arithmetic mean of task queue time plus wallclock time, in seconds (i.e. the amount of time which passes between a task being submitted, and its termination).

Last Job Finished: The time, in seconds, after submission at which the longest-running task terminates.

Times in the summary rows only include those tasks which completed successfully.

Table E.1 – WMS-style submission using type 1 jobs (1,000 tasks).

ID	Resource	S	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.wmin.ac.uk	100.0%	1998.79	6446.27	11296.08
2	miffy.elec.gla.ac.uk	100.0%	2160.97	3681.52	5788.77
3	ngs.manchester.ac.uk	100.0%	2001.89	6437.65	11170.57
4	ngs.leeds.ac.uk	100.0%	1976.76	6306.56	10609.46
5	ngs.manchester.ac.uk	100.0%	1997.58	6407.45	12337.44
6	ngs.wmin.ac.uk	100.0%	1998.18	6363.69	10707.18
7	ngs.wmin.ac.uk	100.0%	1987.30	6479.39	10873.36
8	conan.elec.gla.ac.uk	100.0%	1933.04	3599.58	5857.49
9	scotgrid.ac.uk	100.0%	1986.30	2990.38	4238.34
10	ngs.lancaster.ac.uk	100.0%	2284.48	3476.30	6238.55
11	ngs.oxford.ac.uk	100.0%	1997.27	6389.68	10966.10
12	scotgrid.ac.uk	100.0%	1991.43	2990.36	4686.71
13	ngs.lancaster.ac.uk	100.0%	2293.40	3469.57	5597.65
14	ngs.lancaster.ac.uk	100.0%	2283.13	3479.74	5661.22
15	ngs.wmin.ac.uk	100.0%	1989.90	6419.06	12273.96
16	ngs.glasgow.ac.uk	100.0%	1920.45	5908.62	10447.02
17	ngs.ral.ac.uk	100.0%	2155.93	6384.17	10731.17
18	ngs.ral.ac.uk	100.0%	2161.84	6414.28	10635.38
19	miffy.elec.gla.ac.uk	100.0%	2162.36	3728.19	5416.78
20	ngs.glasgow.ac.uk	100.0%	1931.40	5968.63	10207.54
	SUMMARY	100.0%	2060.62	5167.05	8787.04

Table E.2 – WMS-style submission using type 2 jobs (1,000 tasks).

ID	Resource	S	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	miffy.elec.gla.ac.uk	100.0%	12583.35	14845.32	88071.81
2	scotgrid.ac.uk	100.0%	11274.89	13565.42	83104.60
3	ngs.wmin.ac.uk	100.0%	11091.45	31420.95	83522.54
4	conan.elec.gla.ac.uk	100.0%	11046.14	16326.54	87239.39
5	ngs.glasgow.ac.uk	100.0%	10620.03	29224.36	90212.94
6	conan.elec.gla.ac.uk	100.0%	11115.88	16511.10	76122.99
7	miffy.elec.gla.ac.uk	100.0%	12361.73	17236.62	88161.35
8	ngs.glasgow.ac.uk	100.0%	10765.17	28480.62	109315.34
9	ngs.ral.ac.uk	100.0%	12103.97	30752.19	114294.28
10	ngs.glasgow.ac.uk	100.0%	10616.49	28762.13	94820.70
11	conan.elec.gla.ac.uk	100.0%	10613.98	15836.61	79379.00
12	ngs.oxford.ac.uk	100.0%	11451.86	31849.46	113524.18
13	ngs.leeds.ac.uk	100.0%	11391.63	31863.90	121263.48
14	ngs.lancaster.ac.uk	100.0%	12516.47	15492.60	83768.33
15	conan.elec.gla.ac.uk	100.0%	10513.22	15861.48	82594.87
16	ngs.ral.ac.uk	100.0%	12233.07	31536.99	123869.20
17	conan.elec.gla.ac.uk	100.0%	11337.91	16823.50	85254.45
18	ngs.ral.ac.uk	100.0%	11914.58	30020.79	116681.66
19	conan.elec.gla.ac.uk	100.0%	11019.82	16594.56	86587.70
20	ngs.ral.ac.uk	100.0%	11729.24	30779.40	100485.70
	SUMMARY	100.0%	11415.04	23189.23	95413.73

Table E.3 – IRB-style submission using type 1 jobs (1,000 tasks).

ID	Resource	S	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	1946.80	3372.93	5436.78
2	Various	100.0%	1922.03	3304.05	5355.38
3	Various	100.0%	1943.00	3358.17	5375.32
4	Various	100.0%	1924.83	3302.39	5275.44
5	Various	100.0%	1943.01	3349.57	5327.26
6	Various	100.0%	1935.42	3364.06	5111.98
7	Various	100.0%	1941.99	3341.91	6109.53
8	Various	100.0%	1939.39	3337.10	4862.74
9	Various	100.0%	1957.80	3383.48	6664.58
10	Various	100.0%	1943.77	3340.32	4982.26
11	Various	100.0%	1949.03	3355.96	5240.60
12	Various	100.0%	1916.00	3310.53	5201.01
13	Various	100.0%	1932.34	3349.00	5891.69
14	Various	100.0%	1936.24	3333.61	5844.65
15	Various	100.0%	1925.61	3327.24	4908.66
16	Various	100.0%	1929.99	3370.15	5849.84
17	Various	100.0%	1945.78	3350.53	6448.22
18	Various	100.0%	1953.85	3328.10	5578.04
19	Various	100.0%	1952.18	3364.38	5083.15
20	Various	100.0%	1937.69	3333.15	6311.29
	SUMMARY	100.0%	1938.84	3343.83	5542.92

Table E.4 – IRB-style submission using type 2 jobs (1,000 tasks).

ID	Resource	S	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	11215.57	15132.66	75739.31
2	Various	100.0%	10591.84	14489.26	84378.89
3	Various	100.0%	11074.84	14801.19	74314.69
4	Various	100.0%	11149.07	14948.98	80920.36
5	Various	100.0%	10925.28	14714.39	85651.59
6	Various	100.0%	11062.80	15024.78	84360.38
7	Various	100.0%	11417.90	15269.83	80982.65
8	Various	100.0%	10711.97	14499.92	84507.74
9	Various	100.0%	10997.57	14804.27	82389.90
10	Various	100.0%	11194.30	15056.46	84017.61
11	Various	100.0%	11207.20	15091.96	73371.34
12	Various	100.0%	10746.86	14584.94	84403.93
13	Various	100.0%	11398.23	15268.03	77479.62
14	Various	100.0%	11567.95	15436.26	80384.23
15	Various	100.0%	11061.39	14982.06	79277.87
16	Various	100.0%	11274.72	15022.67	80792.35
17	Various	100.0%	10919.24	14869.49	82852.87
18	Various	100.0%	11405.06	15312.20	81454.42
19	Various	100.0%	10314.42	14228.27	80120.32
20	Various	100.0%	10654.22	14553.35	72418.31
	SUMMARY	100.0%	11044.52	14904.55	80490.92

Table E.5 – WMS-style submission using type 1 jobs with licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	scotgrid.ac.uk	100.0%	0.0%	1989.82	2988.83	4770.74
2	ngs.manchester.ac.uk	100.0%	0.0%	1973.05	6359.54	10735.31
3	ngs.glasgow.ac.uk	100.0%	0.0%	1926.48	5947.42	10202.06
4	ngs.lancaster.ac.uk	0.0%	100.0%	0.74	581.73	595.04
5	ngs.ral.ac.uk	0.0%	100.0%	0.86	624.92	667.63
6	miffy.elec.gla.ac.uk	100.0%	0.0%	2160.65	3723.26	5582.10
7	ngs.ral.ac.uk	0.0%	100.0%	0.83	602.45	623.67
8	ngs.oxford.ac.uk	0.0%	100.0%	0.77	621.42	650.22
9	ngs.manchester.ac.uk	100.0%	0.0%	1983.71	6320.48	13576.34
10	ngs.oxford.ac.uk	0.0%	100.0%	0.67	627.69	662.96
11	ngs.oxford.ac.uk	0.0%	100.0%	0.70	622.53	648.40
12	ngs.lancaster.ac.uk	0.0%	100.0%	0.67	575.33	587.06
13	ngs.ral.ac.uk	0.0%	100.0%	1.02	609.24	646.25
14	scotgrid.ac.uk	100.0%	0.0%	1993.80	2998.08	4618.68
15	ngs.oxford.ac.uk	0.0%	100.0%	0.68	643.44	673.08
16	ngs.glasgow.ac.uk	100.0%	0.0%	1915.86	5914.90	9818.36
17	ngs.lancaster.ac.uk	0.0%	100.0%	0.70	579.75	592.02
18	ngs.leeds.ac.uk	0.0%	100.0%	0.68	589.52	608.25
19	ngs.oxford.ac.uk	0.0%	100.0%	0.69	622.76	653.34
20	scotgrid.ac.uk	100.0%	0.0%	2000.49	3014.99	6478.06
	SUMMARY	40.0%	60.0%	1992.98	4658.44	8221.20

Table E.6 – WMS-style submission using type 2 jobs with licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.leeds.ac.uk	0.0%	100.0%	1.38	815.22	950.10
2	ngs.lancaster.ac.uk	0.0%	100.0%	0.67	599.36	623.23
3	ngs.ral.ac.uk	0.0%	100.0%	1.04	682.43	739.19
4	ngs.ral.ac.uk	0.0%	100.0%	0.72	680.38	736.47
5	miffy.elec.gla.ac.uk	100.0%	0.0%	11763.52	16483.24	86795.99
6	miffy.elec.gla.ac.uk	100.0%	0.0%	12096.14	16725.14	86978.70
7	ngs.manchester.ac.uk	100.0%	0.0%	11167.13	31058.71	111366.87
8	ngs.leeds.ac.uk	0.0%	100.0%	0.76	650.47	695.26
9	conan.elec.gla.ac.uk	100.0%	0.0%	10704.12	16222.07	87900.26
10	ngs.wmin.ac.uk	0.0%	100.0%	0.76	728.99	785.40
11	ngs.oxford.ac.uk	0.0%	100.0%	0.93	785.79	848.60
12	ngs.glasgow.ac.uk	100.0%	0.0%	10524.83	28044.16	106811.23
13	ngs.wmin.ac.uk	0.0%	100.0%	0.73	684.22	742.58
14	ngs.wmin.ac.uk	0.0%	100.0%	0.70	668.44	723.34
15	ngs.wmin.ac.uk	0.0%	100.0%	0.80	735.73	802.30
16	ngs.glasgow.ac.uk	100.0%	0.0%	10876.69	28971.30	103903.79
17	miffy.elec.gla.ac.uk	100.0%	0.0%	11345.42	16104.13	82833.18
18	miffy.elec.gla.ac.uk	100.0%	0.0%	12267.76	16958.16	92234.06
19	miffy.elec.gla.ac.uk	100.0%	0.0%	12431.97	17256.17	95296.69
20	ngs.glasgow.ac.uk	100.0%	0.0%	10271.80	27833.04	94948.03
	SUMMARY	50.0%	50.0%	11344.94	21565.61	94905.88

Table E.7 – WMS-style submission using type 1 jobs with further licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.ral.ac.uk	0.0%	100.0%	0.69	603.83	627.12
2	ngs.wmin.ac.uk	0.0%	100.0%	0.72	623.23	649.06
3	ngs.oxford.ac.uk	0.0%	100.0%	0.53	607.53	629.58
4	ngs.leeds.ac.uk	0.0%	100.0%	0.72	614.69	641.05
5	ngs.oxford.ac.uk	0.0%	100.0%	0.69	643.26	679.94
6	ngs.wmin.ac.uk	0.0%	100.0%	0.54	622.88	650.22
7	ngs.lancaster.ac.uk	0.0%	100.0%	0.69	585.25	597.75
8	ngs.wmin.ac.uk	0.0%	100.0%	0.70	638.29	675.97
9	conan.elec.gla.ac.uk	30.0%	70.0%	581.27	1737.32	4634.57
10	miffy.elec.gla.ac.uk	30.0%	70.0%	652.25	1753.12	4566.26
11	conan.elec.gla.ac.uk	30.0%	70.0%	589.08	1735.85	4045.73
12	ngs.leeds.ac.uk	0.0%	100.0%	1.07	600.55	626.81
13	scotgrid.ac.uk	30.0%	70.0%	596.41	1391.15	5612.22
14	ngs.leeds.ac.uk	0.0%	100.0%	1.35	620.18	652.25
15	conan.elec.gla.ac.uk	30.0%	70.0%	586.88	1674.62	3995.97
16	ngs.lancaster.ac.uk	0.0%	100.0%	0.69	585.15	597.02
17	ngs.manchester.ac.uk	100.0%	0.0%	1983.10	6363.36	10905.13
18	ngs.glasgow.ac.uk	100.0%	0.0%	1918.50	6007.41	10152.84
19	ngs.ral.ac.uk	0.0%	100.0%	0.81	630.37	660.94
20	miffy.elec.gla.ac.uk	30.0%	70.0%	649.84	1668.37	4239.75
	SUMMARY	19.0%	81.0%	1988.14	4601.53	6017.56

Table E.8 – IRB-style submission using type 1 jobs with licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	0.0%	1948.34	3325.49	4967.39
2	Various	100.0%	0.0%	1931.54	3302.53	5578.06
3	Various	100.0%	0.0%	1949.55	3354.30	6718.92
4	Various	100.0%	0.0%	1941.77	3332.01	4983.36
5	Various	100.0%	0.0%	1926.89	3334.58	5879.48
6	Various	100.0%	0.0%	1947.14	3363.37	5776.66
7	Various	100.0%	0.0%	1931.64	3321.29	4896.71
8	Various	100.0%	0.0%	1922.28	3321.60	5472.06
9	Various	100.0%	0.0%	1949.38	3363.10	5330.98
10	Various	100.0%	0.0%	1938.30	3313.27	5213.34
11	Various	100.0%	0.0%	1955.54	3327.20	5165.10
12	Various	100.0%	0.0%	1921.86	3342.33	4769.80
13	Various	100.0%	0.0%	1959.61	3340.90	5466.03
14	Various	100.0%	0.0%	1930.39	3317.39	5648.35
15	Various	100.0%	0.0%	1952.23	3381.39	5581.18
16	Various	100.0%	0.0%	1937.32	3356.49	5016.25
17	Various	100.0%	0.0%	1926.27	3349.94	4983.23
18	Various	100.0%	0.0%	1965.17	3346.25	5014.75
19	Various	100.0%	0.0%	1946.33	3338.26	5160.49
20	Various	100.0%	0.0%	1926.79	3313.86	5161.02
	SUMMARY	100.0%	0.0%	1940.42	3337.28	5339.16

Table E.9 – IRB-style submission using type 2 jobs with licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	0.0%	10704.73	14578.86	82652.50
2	Various	100.0%	0.0%	10667.93	14444.45	55629.99
3	Various	100.0%	0.0%	11311.67	15137.59	79502.56
4	Various	100.0%	0.0%	10346.43	14192.72	74830.37
5	Various	100.0%	0.0%	10829.05	14546.40	76233.99
6	Various	100.0%	0.0%	10308.75	14036.52	75077.45
7	Various	100.0%	0.0%	10339.48	14020.16	51690.13
8	Various	100.0%	0.0%	11162.32	15008.22	75696.24
9	Various	100.0%	0.0%	10810.69	14631.16	78933.17
10	Various	100.0%	0.0%	11140.55	15181.84	85463.62
11	Various	100.0%	0.0%	10845.91	14623.26	83693.48
12	Various	100.0%	0.0%	10589.28	14352.76	76415.42
13	Various	100.0%	0.0%	11358.54	15300.53	83585.45
14	Various	100.0%	0.0%	10891.47	14912.93	86337.04
15	Various	100.0%	0.0%	10795.70	14377.48	76505.87
16	Various	100.0%	0.0%	11591.34	15593.79	84109.04
17	Various	100.0%	0.0%	11236.34	15262.67	85968.99
18	Various	100.0%	0.0%	10570.39	14465.96	85069.22
19	Various	100.0%	0.0%	10934.13	14677.54	85559.86
20	Various	100.0%	0.0%	10788.77	14600.48	81967.76
	SUMMARY	100.0%	0.0%	10861.17	14697.27	78246.11

Table E.10 – IRB-style submission using type 1 jobs with further licence restrictions (1,000 tasks).

ID	Resource	S	L	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	0.0%	1949.67	3108.42	5025.47
2	Various	100.0%	0.0%	1976.24	3123.71	5396.70
3	Various	100.0%	0.0%	1965.00	3105.26	5010.02
4	Various	100.0%	0.0%	1983.43	3105.07	5225.33
5	Various	100.0%	0.0%	1954.53	3099.05	5598.72
6	Various	100.0%	0.0%	1951.79	3099.01	5386.62
7	Various	100.0%	0.0%	1952.21	3102.36	5504.76
8	Various	100.0%	0.0%	1958.63	3108.04	5081.46
9	Various	100.0%	0.0%	1965.44	3139.68	5180.78
10	Various	100.0%	0.0%	1961.24	3121.96	5642.77
11	Various	100.0%	0.0%	1963.56	3099.95	6183.86
12	Various	100.0%	0.0%	1980.06	3104.46	4964.95
13	Various	100.0%	0.0%	1965.12	3100.77	5127.22
14	Various	100.0%	0.0%	1962.41	3111.88	5468.74
15	Various	100.0%	0.0%	1961.29	3069.57	5042.60
16	Various	100.0%	0.0%	1981.91	3123.04	5250.36
17	Various	100.0%	0.0%	1951.55	3092.46	6041.10
18	Various	100.0%	0.0%	1955.06	3103.68	5519.38
19	Various	100.0%	0.0%	1964.26	3090.51	4987.70
20	Various	100.0%	0.0%	1957.75	3105.60	5105.18
	SUMMARY	100.0%	0.0%	1963.06	3105.72	5337.19

Table E.11 – WMS-style submission using type 1 jobs with deadline (1,000 tasks).

ID	Resource	S	DL	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.glasgow.ac.uk	92.7%	7.3%	1925.03	5949.72	10387.24
2	ngs.lancaster.ac.uk	100.0%	0.0%	2311.71	3508.81	6665.58
3	miffy.elec.gla.ac.uk	100.0%	0.0%	2146.05	3709.23	5699.93
4	conan.elec.gla.ac.uk	100.0%	0.0%	1927.78	3608.31	5899.23
5	ngs.manchester.ac.uk	84.3%	15.7%	1999.82	6399.48	11038.36
6	ngs.wmin.ac.uk	83.3%	16.7%	1989.89	6387.27	10683.62
7	ngs.manchester.ac.uk	84.5%	15.5%	1981.41	6361.47	10724.78
8	ngs.leeds.ac.uk	82.6%	17.4%	1988.61	6420.19	10543.58
9	miffy.elec.gla.ac.uk	100.0%	0.0%	2154.92	3691.59	6117.82
10	ngs.glasgow.ac.uk	92.5%	7.5%	1923.52	5981.39	10120.98
11	ngs.manchester.ac.uk	83.0%	17.0%	2005.90	6394.13	10791.41
12	ngs.lancaster.ac.uk	100.0%	0.0%	2276.12	3470.16	5163.54
13	ngs.glasgow.ac.uk	93.6%	6.4%	1909.24	5925.84	10305.38
14	ngs.oxford.ac.uk	82.8%	17.2%	1992.22	6466.90	11172.52
15	miffy.elec.gla.ac.uk	100.0%	0.0%	2160.06	3723.96	5802.05
16	conan.elec.gla.ac.uk	100.0%	0.0%	1936.53	3628.77	6049.27
17	ngs.wmin.ac.uk	82.1%	17.9%	1998.21	6452.63	10908.29
18	ngs.leeds.ac.uk	83.7%	16.3%	1990.91	6379.60	10612.63
19	ngs.wmin.ac.uk	82.5%	17.5%	2007.91	6479.02	10976.52
20	ngs.ral.ac.uk	81.8%	18.2%	2184.04	6535.79	11607.30
	SUMMARY	90.5%	9.5%	2035.84	4924.87	7915.51

Table E.12 – WMS-style submission using type 2 jobs with deadline (1,000 tasks).

ID	Resource	S	DL	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.glasgow.ac.uk	85.9%	14.1%	11192.02	29225.05	110335.06
2	ngs.glasgow.ac.uk	90.2%	9.8%	10474.33	27617.34	100906.26
3	ngs.oxford.ac.uk	75.7%	24.3%	11456.39	31993.96	112189.39
4	conan.elec.gla.ac.uk	99.3%	0.7%	10586.67	15915.53	74926.09
5	ngs.ral.ac.uk	80.7%	19.3%	12127.56	30769.41	100306.62
6	conan.elec.gla.ac.uk	99.3%	0.7%	10666.48	16171.59	82731.91
7	ngs.glasgow.ac.uk	87.5%	12.5%	10832.03	28511.02	99675.04
8	ngs.wmin.ac.uk	77.2%	22.8%	11527.44	31385.63	113382.06
9	miffy.elec.gla.ac.uk	98.5%	1.5%	12582.93	17438.61	87026.79
10	ngs.lancaster.ac.uk	99.1%	0.9%	13068.19	16091.85	96120.62
11	scotgrid.ac.uk	98.9%	1.1%	10884.22	13193.85	80028.91
12	ngs.ral.ac.uk	82.2%	17.8%	11743.86	30706.66	108645.40
13	ngs.oxford.ac.uk	78.1%	21.9%	11143.67	31602.30	111579.78
14	ngs.glasgow.ac.uk	87.9%	12.1%	10477.18	28415.66	101808.80
15	scotgrid.ac.uk	98.8%	1.2%	10944.57	13174.02	80976.66
16	conan.elec.gla.ac.uk	98.5%	1.5%	11023.72	16403.12	81993.98
17	ngs.oxford.ac.uk	78.8%	21.2%	10747.95	30544.90	76896.21
18	ngs.manchester.ac.uk	79.8%	20.2%	10881.56	31096.16	101424.39
19	ngs.leeds.ac.uk	81.9%	18.1%	10931.76	30241.20	111739.37
20	ngs.oxford.ac.uk	77.3%	22.7%	11312.60	31978.02	106674.77
	SUMMARY	87.8%	12.2%	10061.51	21446.02	39811.77

Table E.13 – IRB-style submission using type 1 jobs with deadline (1,000 tasks).

ID	Resource	S	DL	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	0.0%	1977.20	3319.38	5531.32
2	Various	100.0%	0.0%	1994.74	3331.72	6788.18
3	Various	100.0%	0.0%	1990.77	3276.17	5191.51
4	Various	100.0%	0.0%	1988.42	3291.80	5543.86
5	Various	100.0%	0.0%	1983.78	3306.44	6240.60
6	Various	100.0%	0.0%	1974.17	3283.61	5085.02
7	Various	100.0%	0.0%	1956.89	3271.35	7183.97
8	Various	100.0%	0.0%	1999.33	3332.45	5161.74
9	Various	100.0%	0.0%	1978.96	3287.39	5560.38
10	Various	100.0%	0.0%	1974.00	3297.27	5368.33
11	Various	100.0%	0.0%	1989.77	3286.91	5352.78
12	Various	100.0%	0.0%	1990.04	3308.71	5233.55
13	Various	100.0%	0.0%	1967.27	3267.60	5075.30
14	Various	100.0%	0.0%	1980.22	3325.93	4906.74
15	Various	100.0%	0.0%	1982.68	3287.99	5828.86
16	Various	100.0%	0.0%	1983.46	3304.23	5245.77
17	Various	100.0%	0.0%	1961.89	3283.98	5264.10
18	Various	100.0%	0.0%	1988.99	3309.86	5306.42
19	Various	100.0%	0.0%	1988.38	3318.28	5440.87
20	Various	100.0%	0.0%	1961.47	3273.29	5573.14
	SUMMARY	100.0%	0.0%	1980.62	3298.22	5544.12

Table E.14 – IRB-style submission using type 2 jobs with deadline (1,000 tasks).

ID	Resource	S	DL	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.3%	0.7%	10778.08	14871.47	77610.63
2	Various	99.1%	0.9%	10817.92	15015.53	74919.29
3	Various	98.4%	1.6%	11083.75	15330.74	76991.96
4	Various	98.5%	1.5%	11281.51	15473.95	85656.40
5	Various	99.1%	0.9%	10753.16	15100.28	77228.21
6	Various	98.6%	1.4%	11115.18	15162.34	84388.56
7	Various	98.3%	1.7%	11041.23	15118.91	85583.19
8	Various	98.9%	1.1%	10982.05	15342.55	79272.59
9	Various	99.6%	0.4%	10120.27	14283.99	71992.04
10	Various	99.0%	1.0%	10571.56	14681.59	80467.91
11	Various	98.4%	1.6%	11444.97	15804.77	85705.64
12	Various	99.3%	0.7%	10687.18	14794.73	68683.75
13	Various	98.8%	1.2%	11023.06	15244.22	82665.26
14	Various	99.0%	1.0%	10898.46	15014.53	82626.94
15	Various	99.1%	0.9%	10772.26	14945.24	79405.05
16	Various	98.4%	1.6%	11294.01	15371.26	81319.54
17	Various	98.7%	1.3%	10744.14	14913.41	80942.73
18	Various	98.6%	1.4%	10915.34	15104.89	83309.37
19	Various	99.1%	0.9%	10985.63	15251.77	85631.03
20	Various	98.7%	1.3%	11168.88	15257.70	78903.69
	SUMMARY	98.8%	1.2%	10392.92	14572.73	33153.65

Table E.15 – WMS-style submission using type 1 jobs with downtime (1,000 tasks).

ID	Resource	S	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.oxford.ac.uk	75.4%	24.6%	1990.44	5385.77	9594.81
2	ngs.leeds.ac.uk	100.0%	0.0%	1989.78	5371.97	9540.23
3	ngs.ral.ac.uk	28.8%	71.2%	2167.63	5364.14	10014.72
4	miffy.elec.gla.ac.uk	100.0%	0.0%	2181.94	2685.52	4636.71
5	ngs.wmin.ac.uk	100.0%	0.0%	2014.84	5446.19	9724.53
6	ngs.glasgow.ac.uk	100.0%	0.0%	1934.13	5023.36	8950.90
7	miffy.elec.gla.ac.uk	100.0%	0.0%	2143.63	2647.21	4865.98
8	ngs.lancaster.ac.uk	100.0%	0.0%	2300.04	2803.62	6538.19
9	scotgrid.ac.uk	100.0%	0.0%	1994.09	2497.67	3767.93
10	ngs.ral.ac.uk	28.9%	71.1%	2177.71	5377.50	9966.38
11	ngs.leeds.ac.uk	100.0%	0.0%	1999.71	5413.13	9724.62
12	miffy.elec.gla.ac.uk	100.0%	0.0%	2177.80	2681.38	4845.94
13	ngs.glasgow.ac.uk	100.0%	0.0%	1918.46	4992.82	8817.21
14	scotgrid.ac.uk	100.0%	0.0%	2001.38	2504.96	3813.22
15	ngs.wmin.ac.uk	100.0%	0.0%	1993.49	5405.04	9970.50
16	ngs.glasgow.ac.uk	100.0%	0.0%	1932.94	5024.78	9749.30
17	ngs.wmin.ac.uk	100.0%	0.0%	1981.14	5338.09	9407.83
18	conan.elec.gla.ac.uk	1.2%	98.8%	1933.52	2717.03	5640.30
19	ngs.lancaster.ac.uk	100.0%	0.0%	2301.74	2805.32	6539.11
20	ngs.lancaster.ac.uk	100.0%	0.0%	2293.10	2796.68	6232.22
	SUMMARY	86.7%	13.3%	2070.94	3940.49	6669.62

Table E.16 – WMS-style submission using type 2 jobs with downtime (1,000 tasks).

ID	Resource	S	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.lancaster.ac.uk	99.3%	0.7%	12891.38	13394.96	91626.98
2	ngs.ral.ac.uk	58.6%	41.4%	12091.48	25619.43	107222.83
3	ngs.oxford.ac.uk	56.3%	43.7%	11627.17	26765.55	95402.09
4	miffy.elec.gla.ac.uk	100.0%	0.0%	12053.37	12556.95	84117.17
5	ngs.leeds.ac.uk	99.9%	0.0%	11477.02	26882.59	106958.62
6	ngs.wmin.ac.uk	99.9%	0.0%	11007.59	25141.42	89535.54
7	conan.elec.gla.ac.uk	98.8%	1.2%	10915.63	12256.95	77760.39
8	ngs.lancaster.ac.uk	98.2%	1.8%	13364.31	13867.89	91625.74
9	conan.elec.gla.ac.uk	98.8%	1.2%	10981.42	12274.45	77752.94
10	ngs.manchester.ac.uk	99.8%	0.0%	11979.13	27597.37	115486.52
11	ngs.ral.ac.uk	60.4%	39.6%	11926.60	25283.86	93831.32
12	ngs.oxford.ac.uk	57.4%	42.6%	11116.09	25897.72	98544.21
13	ngs.oxford.ac.uk	59.3%	40.7%	11046.23	25604.07	92033.75
14	ngs.lancaster.ac.uk	99.0%	1.0%	13142.67	13646.25	91625.82
15	ngs.leeds.ac.uk	99.7%	0.0%	11260.88	25180.90	92825.42
16	scotgrid.ac.uk	0.0%	100.0%	11298.64	11802.22	79712.30
17	ngs.glasgow.ac.uk	99.7%	0.0%	10499.42	23067.55	102073.70
18	ngs.glasgow.ac.uk	99.9%	0.0%	10755.16	23768.33	94985.74
19	ngs.manchester.ac.uk	99.6%	0.0%	11097.38	25523.79	103149.03
20	ngs.oxford.ac.uk	57.4%	42.6%	11326.33	25950.88	106327.62
	SUMMARY	82.2%	17.8%	10915.64	18652.55	52018.52

Table E.17 – IRB-style submission using type 1 jobs with downtime (1,000 tasks).

ID	Resource	S	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.9%	0.1%	2001.10	2902.87	4424.68
2	Various	99.8%	0.2%	2019.17	2954.91	5000.16
3	Various	100.0%	0.0%	2006.47	2909.90	6091.45
4	Various	99.9%	0.1%	1980.62	2885.25	4457.31
5	Various	99.8%	0.2%	1996.13	2926.70	5222.94
6	Various	100.0%	0.0%	1989.87	2890.97	5476.64
7	Various	99.5%	0.5%	2003.91	2914.28	4656.19
8	Various	99.6%	0.4%	2011.08	2936.98	4978.88
9	Various	99.5%	0.5%	1999.77	2922.98	4478.58
10	Various	99.9%	0.1%	2017.61	2936.75	6208.46
11	Various	99.6%	0.4%	1989.48	2890.79	4959.64
12	Various	99.5%	0.5%	2020.07	2931.12	4475.58
13	Various	99.6%	0.4%	2017.40	2943.06	5219.18
14	Various	99.5%	0.5%	2008.72	2928.48	5847.03
15	Various	99.2%	0.8%	1997.73	2922.73	5048.31
16	Various	99.4%	0.6%	2003.89	2902.04	5311.88
17	Various	98.8%	1.2%	2013.39	2927.31	6014.40
18	Various	98.9%	1.1%	2007.20	2911.71	5226.55
19	Various	98.2%	1.8%	2015.76	2941.47	5461.80
20	Various	99.1%	0.9%	1997.19	2916.71	5345.31
	SUMMARY	99.5%	0.5%	2001.29	2916.76	5143.92

Table E.18 – IRB-style submission using type 1 jobs with downtime (1,000 tasks, DMO).

ID	Resource	S	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.9%	0.1%	2003.65	2507.23	5089.48
2	Various	99.8%	0.2%	1978.87	2482.45	4585.57
3	Various	100.0%	0.0%	1991.18	2494.76	4213.71
4	Various	99.9%	0.1%	1971.86	2475.44	4098.34
5	Various	100.0%	0.0%	1995.30	2498.88	4636.66
6	Various	100.0%	0.0%	1982.62	2486.20	3762.13
7	Various	99.9%	0.1%	1987.92	2491.50	3842.83
8	Various	99.7%	0.3%	1977.44	2481.02	4599.18
9	Various	99.9%	0.1%	1989.53	2493.11	4284.45
10	Various	100.0%	0.0%	1982.60	2486.18	3807.65
11	Various	99.9%	0.1%	1989.16	2492.74	4162.40
12	Various	99.9%	0.1%	1997.67	2501.25	4169.89
13	Various	99.8%	0.2%	1988.63	2492.21	4429.02
14	Various	99.7%	0.3%	1995.45	2499.03	4428.70
15	Various	100.0%	0.0%	1985.57	2489.15	3719.38
16	Various	99.7%	0.3%	1985.46	2489.04	4638.20
17	Various	100.0%	0.0%	1993.69	2497.27	3903.59
18	Various	99.7%	0.3%	1987.77	2491.35	4278.58
19	Various	100.0%	0.0%	1975.81	2479.39	4161.70
20	Various	99.5%	0.5%	1992.49	2496.07	5360.66
	SUMMARY	99.9%	0.1%	1985.42	2489.00	4022.09

Table E.19 – IRB-style submission using type 2 jobs with downtime (1,000 tasks).

ID	Resource	S	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.0%	1.0%	10855.27	11358.85	77762.02
2	Various	98.9%	1.1%	11023.94	11527.52	77758.73
3	Various	98.6%	1.4%	11401.34	11904.92	77755.49
4	Various	99.7%	0.3%	10230.51	10734.09	76773.32
5	Various	98.5%	1.5%	10846.05	11349.63	77753.42
6	Various	98.9%	1.1%	10615.25	11118.83	77760.62
7	Various	99.3%	0.7%	10769.38	11272.96	77747.58
8	Various	99.1%	0.9%	10552.41	11055.99	77753.25
9	Various	99.0%	1.0%	11040.99	11544.57	77755.22
10	Various	99.5%	0.5%	10734.25	11237.83	77747.86
11	Various	98.4%	1.6%	11430.36	11933.94	77746.61
12	Various	99.2%	0.8%	10318.16	10821.74	72542.73
13	Various	99.4%	0.6%	10620.49	11124.07	77752.08
14	Various	98.4%	1.6%	11120.36	11623.94	75779.34
15	Various	98.9%	1.1%	10888.17	11391.75	77749.85
16	Various	99.1%	0.9%	10850.13	11353.71	77750.69
17	Various	99.4%	0.6%	10818.99	11322.57	77746.22
18	Various	99.2%	0.8%	10869.93	11373.51	77752.06
19	Various	99.1%	0.9%	10812.31	11315.89	75778.80
20	Various	98.9%	1.1%	10602.96	11106.54	77754.94
	SUMMARY	99.0%	1.0%	10375.66	10879.23	65456.47

Table E.20 – WMS-style submission using type 2 jobs with combined requirements (1,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.wmin.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
2	scotgrid.ac.uk	50.4%	49.6%	0.0%	0.0%	11392.26	12227.48	89613.86
3	ngs.manchester.ac.uk	24.7%	75.3%	0.0%	0.0%	11948.60	16085.03	84276.20
4	scotgrid.ac.uk	50.3%	49.7%	0.0%	0.0%	11654.20	12514.73	125562.58
5	conan.elec.gla.ac.uk	51.7%	48.1%	0.0%	0.2%	10383.63	12810.66	46837.62
6	miffy.elec.gla.ac.uk	50.9%	48.3%	0.0%	0.8%	11681.24	13714.52	46272.74
7	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
8	scotgrid.ac.uk	50.4%	49.6%	0.0%	0.0%	10089.47	10895.20	122249.91
9	miffy.elec.gla.ac.uk	51.1%	48.5%	0.0%	0.4%	11471.18	13654.79	48927.74
10	conan.elec.gla.ac.uk	51.7%	47.6%	0.0%	0.7%	10490.11	12827.73	49672.15
11	ngs.manchester.ac.uk	23.1%	76.9%	0.0%	0.0%	10845.41	14538.54	137406.41
12	ngs.manchester.ac.uk	25.4%	74.6%	0.0%	0.0%	11575.25	16030.08	135206.46
13	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
14	miffy.elec.gla.ac.uk	50.5%	48.8%	0.0%	0.7%	11567.17	13728.79	37965.76
15	ngs.manchester.ac.uk	25.4%	74.6%	0.0%	0.0%	11172.28	14948.57	88402.66
16	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
17	miffy.elec.gla.ac.uk	51.2%	48.3%	0.0%	0.5%	11704.98	13770.33	59771.40
18	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
19	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
20	ngs.leeds.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
	SUMMARY	27.8%	72.0%	0.0%	0.2%	11199.16	13352.28	49937.19

Table E.21 – IRB-style submission using type 2 jobs with combined requirements (1,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.4%	0.0%	0.0%	0.6%	10355.79	12742.13	56234.28
2	Various	99.1%	0.0%	0.0%	0.9%	10630.67	13044.93	95421.75
3	Various	99.1%	0.0%	0.0%	0.9%	10169.72	12511.14	85005.23
4	Various	98.8%	0.0%	0.0%	1.2%	10687.78	12947.74	87444.65
5	Various	99.4%	0.0%	0.0%	0.6%	10990.59	13516.41	159047.98
6	Various	98.9%	0.0%	0.0%	1.1%	10978.54	13246.47	120314.66
7	Various	98.7%	0.0%	0.0%	1.3%	10520.47	12822.32	89421.72
8	Various	98.8%	0.0%	0.0%	1.2%	10914.24	13412.70	123394.67
9	Various	98.6%	0.0%	0.0%	1.4%	10336.30	12802.31	82167.30
10	Various	99.3%	0.0%	0.0%	0.7%	10619.89	12920.98	88420.21
11	Various	98.6%	0.0%	0.0%	1.4%	10178.35	12651.34	130768.61
12	Various	99.2%	0.0%	0.0%	0.8%	10779.25	13061.02	85763.98
13	Various	99.3%	0.0%	0.0%	0.7%	10553.11	12976.57	85878.67
14	Various	99.1%	0.0%	0.0%	0.9%	10631.49	12903.61	137351.67
15	Various	98.3%	0.0%	0.0%	1.7%	10734.42	13122.50	117151.10
16	Various	99.1%	0.0%	0.0%	0.9%	10914.50	13385.01	120352.68
17	Various	98.9%	0.0%	0.0%	1.1%	10778.71	13062.77	82483.90
18	Various	98.8%	0.0%	0.0%	1.2%	10646.91	12964.74	86017.36
19	Various	98.8%	0.0%	0.0%	1.2%	10498.73	12777.09	87680.85
20	Various	98.9%	0.0%	0.0%	1.1%	10473.79	12763.87	86765.15
	SUMMARY	99.0%	0.0%	0.0%	1.0%	10619.76	12981.89	79772.67

Table E.22 – WMS-style submission using type 2 jobs with combined requirements (2,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
2	ngs.glasgow.ac.uk	2.5%	97.5%	0.0%	0.0%	9407.31	10058.64	27594.07
3	miffy.elec.gla.ac.uk	25.7%	74.1%	0.0%	0.2%	11601.68	13742.05	33457.72
4	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
5	scotgrid.ac.uk	25.3%	74.7%	0.0%	0.0%	11311.93	12171.94	153468.59
6	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
7	ngs.leeds.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
8	conan.elec.gla.ac.uk	26.1%	73.5%	0.0%	0.5%	9865.35	12343.92	60580.94
9	conan.elec.gla.ac.uk	26.2%	73.3%	0.0%	0.5%	10105.54	12434.12	55885.66
10	miffy.elec.gla.ac.uk	25.7%	74.0%	0.0%	0.3%	11407.48	13546.49	39937.25
11	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
12	ngs.leeds.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
13	ngs.oxford.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
14	ngs.manchester.ac.uk	13.1%	86.9%	0.0%	0.0%	10265.54	14320.79	91887.28
15	scotgrid.ac.uk	25.4%	74.6%	0.0%	0.0%	11590.72	12380.74	160243.26
16	ngs.oxford.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
17	ngs.oxford.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
18	ngs.glasgow.ac.uk	2.5%	97.5%	0.0%	0.0%	9543.17	10154.59	30201.08
19	scotgrid.ac.uk	25.4%	74.7%	0.0%	0.0%	11097.23	11941.58	94493.67
20	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
	SUMMARY	9.9%	90.0%	0.0%	0.1%	10904.10	12699.07	33298.60

Table E.23 – IRB-style submission using type 2 jobs with combined requirements (2,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	99.9%	0.0%	0.0%	0.1%	11059.80	18804.60	150062.25
2	Various	99.8%	0.0%	0.0%	0.2%	11158.29	18660.00	139556.14
3	Various	99.8%	0.0%	0.0%	0.2%	11177.86	19107.40	194767.76
4	Various	99.5%	0.0%	0.0%	0.5%	11027.41	18759.39	128501.66
5	Various	99.9%	0.0%	0.0%	0.1%	11002.43	18922.86	159806.74
6	Various	99.6%	0.0%	0.0%	0.4%	10874.30	18662.04	142796.39
7	Various	99.7%	0.0%	0.0%	0.3%	10827.32	18651.70	162761.30
8	Various	99.8%	0.0%	0.0%	0.2%	10850.54	18515.68	99554.13
9	Various	99.6%	0.0%	0.0%	0.4%	10886.68	18800.94	99332.78
10	Various	99.9%	0.0%	0.0%	0.1%	10784.72	18409.59	94257.77
11	Various	99.8%	0.0%	0.0%	0.2%	10673.15	18203.35	189068.46
12	Various	99.9%	0.0%	0.0%	0.1%	10801.23	18376.40	118182.58
13	Various	99.6%	0.0%	0.0%	0.4%	10863.52	18561.46	139094.37
14	Various	99.9%	0.0%	0.0%	0.1%	11025.37	18769.04	169695.42
15	Various	99.9%	0.0%	0.0%	0.1%	10573.03	18095.33	90000.44
16	Various	99.8%	0.0%	0.0%	0.2%	10829.72	18602.13	101290.18
17	Various	99.9%	0.0%	0.0%	0.1%	11014.59	18725.68	101838.13
18	Various	99.8%	0.0%	0.0%	0.2%	11086.15	19012.85	133124.50
19	Various	99.7%	0.0%	0.0%	0.4%	10963.69	18761.77	162416.46
20	Various	99.6%	0.0%	0.0%	0.4%	10936.96	18482.57	137994.47
	SUMMARY	99.8%	0.0%	0.0%	0.2%	10920.83	18644.22	92089.19

Table E.24 – WMS-style submission using type 1 jobs with combined requirements (1,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	ngs.wmin.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
2	miffy.elec.gla.ac.uk	78.8%	21.2%	0.0%	0.0%	2166.02	4548.21	8464.20
3	ngs.manchester.ac.uk	58.5%	41.5%	0.0%	0.0%	2005.58	6762.15	17795.45
4	miffy.elec.gla.ac.uk	75.0%	25.0%	0.0%	0.0%	2186.26	4432.41	8902.96
5	miffy.elec.gla.ac.uk	81.4%	18.6%	0.0%	0.0%	2154.65	4618.15	16198.71
6	ngs.leeds.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
7	ngs.wmin.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
8	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
9	ngs.manchester.ac.uk	48.5%	51.5%	0.0%	0.0%	2001.35	6173.03	16574.52
10	ngs.wmin.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
11	conan.elec.gla.ac.uk	98.4%	1.6%	0.0%	0.0%	1936.21	4965.82	15975.70
12	ngs.oxford.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
13	scotgrid.ac.uk	51.6%	48.4%	0.0%	0.0%	2005.59	2811.61	7251.79
14	ngs.lancaster.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
15	ngs.ral.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
16	conan.elec.gla.ac.uk	92.7%	7.3%	0.0%	0.0%	1948.91	4786.07	8526.08
17	ngs.leeds.ac.uk	0.0%	100.0%	0.0%	0.0%	0.00	0.00	0.00
18	conan.elec.gla.ac.uk	94.6%	5.4%	0.0%	0.0%	1946.24	4821.80	16480.97
19	ngs.glasgow.ac.uk	4.8%	95.2%	0.0%	0.0%	1929.59	2386.63	7976.90
20	conan.elec.gla.ac.uk	95.6%	4.4%	0.0%	0.0%	1949.41	4921.52	14234.93
	SUMMARY	39.0%	61.0%	0.0%	0.0%	2024.42	4843.19	6919.11

Table E.25 – IRB-style submission using type 1 jobs with combined requirements (1,000 tasks).

ID	Resource	S	L	DL	DT	Mean Execution Time (s)	Mean Duration (s)	Last Job Finished (s)
1	Various	100.0%	0.0%	0.0%	0.0%	2095.89	4584.84	16394.30
2	Various	100.0%	0.0%	0.0%	0.0%	2080.61	4622.87	16771.67
3	Various	100.0%	0.0%	0.0%	0.0%	2095.24	4734.19	15674.95
4	Various	100.0%	0.0%	0.0%	0.0%	2094.80	4767.48	16870.70
5	Various	100.0%	0.0%	0.0%	0.0%	2095.91	4719.09	15394.31
6	Various	100.0%	0.0%	0.0%	0.0%	2094.49	4787.60	15834.20
7	Various	100.0%	0.0%	0.0%	0.0%	2095.82	4797.68	16841.64
8	Various	100.0%	0.0%	0.0%	0.0%	2087.51	4704.26	15876.40
9	Various	100.0%	0.0%	0.0%	0.0%	2089.07	4861.98	17069.76
10	Various	100.0%	0.0%	0.0%	0.0%	2099.85	4758.62	15360.36
11	Various	100.0%	0.0%	0.0%	0.0%	2098.51	4684.65	16395.73
12	Various	100.0%	0.0%	0.0%	0.0%	2087.55	4615.41	17198.25
13	Various	100.0%	0.0%	0.0%	0.0%	2091.13	4672.75	15067.22
14	Various	100.0%	0.0%	0.0%	0.0%	2088.05	4758.35	16009.91
15	Various	100.0%	0.0%	0.0%	0.0%	2085.96	4713.67	16593.90
16	Various	100.0%	0.0%	0.0%	0.0%	2084.32	4655.33	17319.10
17	Various	100.0%	0.0%	0.0%	0.0%	2086.35	4780.70	15607.56
18	Various	100.0%	0.0%	0.0%	0.0%	2086.53	4710.49	14935.66
19	Various	100.0%	0.0%	0.0%	0.0%	2093.14	4677.72	16786.91
20	Various	100.0%	0.0%	0.0%	0.0%	2077.54	4688.14	15431.06
	SUMMARY	100.0%	0.0%	0.0%	0.0%	2090.41	4714.79	16171.68

List of Abbreviations

A	
AA	Attribute Authority
AC	Attribute Certificate
ACL	Access Control List
AFS	Andrew File System
B	
BDII	Berkeley Database Information Index
C	
CA	Certification Authority
CAS	Community Authorization Service
CMOS	Complementary Metal-Oxide-Semiconductor
D	
DN	Distinguished Name
DRM	Digital Rights Management
E	
EC2	Elastic Compute Cloud
EGEE	Enabling Grids for E-sciencE
EPSRC	Engineering and Physical Sciences Research Council
F	
FTP	File Transfer Protocol
G	
GIIS	Grid Information Index Service
GLUE	Grid Laboratory Uniform Environment
GRAM	Grid Resource Allocation and Management
GSI	Grid Security Infrastructure
GSS	Generic Security Service
GT	Globus Toolkit

H	
HPC	High-Performance Computing
HTC	High-Throughput Computing
I	
IP	Intellectual Property
IRB	Improved Resource Broker
ITRS	International Technology Roadmap for Semiconductors
I-WAY	Information Wide Area Year
J	
JISC	Joint Information Systems Committee
JPL	Job Policy Language
JSDL	Job Submission Description Language
L	
LCG	LHC Computing Grid
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LHC	Large Hadron Collider
LRM	Local Resource Manager
LRMS	Local Resource Management System
LSF	Load Sharing Facility
M	
MPI	Message Passing Interface
MPP	Massively-Parallel Processor
MTC	Many-Task Computing
N	
NCSA	National Center for Supercomputing Applications
NeSC	National e-Science Centre
NFS	Network File System
NGS	National Grid Service
NVM	Non-Volatile Memory
O	
OGSA	Open Grid Services Architecture
OGSI	Open Grid Services Infrastructure

P	
PBS	Portable Batch System
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
POSIX	Portable Operating System Interface
R	
RBAC	Role-Based Access Control
REST	REpresentational State Transfer
S	
SAML	Security Assertion Mark-up Language
SDL	Scheduling Description Language
SLURM	Simple Linux Utility for Resource Management
SPMD	Single Process, Multiple Data
SRB	Storage Resource Broker
T	
TORQUE	Terascale Open-source Resource and QUEue manager
U	
UDDI	Universal Description, Discovery and Integration
UEE	Uniform Execution Environment
URI	Uniform Resource Identifier
V	
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
W	
W3C	World Wide Web Consortium
WMS	Workload Management System
WSDL	Web Services Description Language
WSRF	Web Services Resource Framework
X	
XML	eXtensible Mark-up Language

Bibliography

[1] ISO/IEC IS 10918-1 – ITU-T Recommendation T.81 – Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines. Recommendation, CCITT Study Group VIII and the Joint Photographic Experts Group (JPEG), September 1992.

[2] ISO/IEC IS 11172-1 – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s. International standard, International Organization for Standardization / International Electrotechnical Commission, August 1993.

[3] ISO/IEC IS 9594-8 – ITU-T Recommendation X.509 – Information Technology – Open Systems Interconnection – The Directory: Public-key and Attribute Certificate Frameworks. Recommendation, International Telecommunications Union, August 2005.

[4] International Technology Roadmap for Semiconductors – Executive Summary. Tech. rep., 2007. `http://www.itrs.net/Links/2007ITRS/ExecSum2007.pdf`, retrieved on 04 December 2008.

[5] Connecting People to Resources: Federated Access Management. Briefing Paper Version 3, JISC, September 2008. `http://www.jisc.ac.uk/media/documents/publications/bpfaminstitutionsv3.pdf`, retrieved on 25 March 2010.

[6] Oracle Grid Engine: An Overview. Oracle White Paper, Oracle Corporation, August 2010. `http://www.oracle.com/technetwork/oem/host-server-mgmt/twp-gridengine-overview-167117.pdf`, retrieved on 17 August 2012.

[7] Introduction to OIDs and the OID Resolution System (ORS). Tech. rep., OID Repository, 2012. http://oid-info.com/introduction.htm, retrieved on 15 August 2012.

[8] ABRAMSON, D., GIDDY, J., AND KOTLER, L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In *Proceedings of the 14th International Parallel and Distributed Processing Symposium* (May 2000), pp. 520–528. DOI:10.1109/IPDPS.2000.846030, ISBN 0-7695-0574-0.

[9] ABRAMSON, D., SOSIC, R., GIDDY, J., AND HALL, B. Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations. In *Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing* (August 1995), pp. 112–121. ISBN 0-8186-7088-6.

[10] ADAMS, C., AND LLOYD, S. *Understanding PKI: Concepts, Standards and Deployment Considerations*, third ed. Addison-Wesley, 2002. ISBN 978-0-6723-2391-1.

[11] ALFIERI, R., AREZZINI, S., CIAMPA, A., PIETRI, R. D., AND MAZZONI, E. HPC on the Grid: The Theophys Experience. *Journal of Grid Computing* (August 2012). DOI:10.1007/s10723-012-9223-6, ISSN 1570-7873.

[12] ALFIERI, R., CECCHINI, R., CIASCHINI, V., DELL'AGNELLO, L., FROHNER, Á., GIONALI, A., LÕRENTEY, K., AND SPARTARO, F. *Grid Computing*, vol. 2970/2004 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, March 2004, ch. VOMS, an Authorization System for Virtual Organizations, pp. 33–40. DOI:10.1007/b95647, ISBN 978-3-540-21048-1.

[13] ALFIERI, R., CECCHINI, R., CIASCHINI, V., DELL'AGNELLO, L., FROHNER, Á., LÕRENTEY, K., AND SPARTARO, F. From Gridmap-File to VOMS: Managing Authorization in a Grid Environment. *Future Generation Computer Systems 21*, 4 (April 2005), 549–558. DOI:10.1016/j.future.2004.10.006.

[14] ANDREOZZI, S., EHM, F., FIELD, L., GALANG, G., KONYA, B., LITMAATH, M., MILLAR, P., AND NAVARRO, J. GLUE Specification v. 2.0. Recommendation GFD-R-P.147, Open Grid Forum, March 2009. http://www.ogf.org/documents/GFD.147.pdf, retrieved on 16 June 2010.

[15] ANJOMSHOAA, A. JSDL LSF Extension Schema. Tech. Rep. doc12459, Open Grid Forum, February 2006. https://forge.ogf.org/sf/sfmain/do/go/doc12459, retrieved on 20 April 2012.

[16] ANJOMSHOAA, A., BRISARD, F., DRESCHER, M., FELLOWS, D., LY, A., MCGOUGH, S., PULSIPHER, D., AND SAVVA, A. Job Submission Description Language (JSDL) Specification, Version 1.0. Tech. Rep. GFD-R-P.056, Global Grid Forum, November 2005. http://www.gridforum.org/documents/GFD.56.pdf, retrieved on 12 January 2009.

[17] ARMSTRONG, P. *Building a Scheduler Adapter for the GridWay Metascheduler*. Thesis in partial fullfillment of the requirements of the Bachelor of Science degree, Computer Science, Faculty of Engineering, University of Victoria, Victoria, Canada, August 2006.

[18] ASENOV, A., BROWN, A. R., DAVIES, J. H., KAYA, S., AND SLAVCHEVA, G. Simulation of Intrinsic Parameter Fluctuations in Decananometer and Nanometer-Scale MOSFETs. *IEEE Transactions on Electron Devices 50*, 9 (September 2003), 1837–1852. DOI:10.1109/TED.2003.815862, ISSN 0018-9383.

[19] ASENOV, A., AND SINNOTT, R. Meeting the Design Challenges of Nano-CMOS Electronics. EPSRC Grant Application EP/E003125/1, University of Glasgow, 2006.

[20] AUSTIN, D., BARBIR, A., FERRIS, C., AND GARG, S. Web Services Architecture Requirements. Tech. rep., W3C Working Group, February 2004. http://www.w3.org/TR/wsa-reqs, retrieved on 09 January 2009.

[21] AZZEDIN, F., AND MAHESWARAN, M. Evolving and Managing Trust in Grid Computing Systems. In *Proceedings of the 2002 IEEE Canadian Conference on Electrical and Computer Engineering* (May 2002), W. Kinsner, A. Sebak, and K. Ferens, Eds., vol. 3, pp. 1424–1429. DOI:10.1109/CCECE.2002.1012962, ISBN 0-7803-7514-9.

[22] BAINBRIDGE, D. The Gowers Review of Intellectual Property. *Intellectual Property & IT Law 11*, 6 (January 2007), 2–7.

[23] BANKS, T. Web Services Resource Framework (WSRF) – Primer v1.2. Tech. rep., OASIS, May 2006. http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf, retrieved on 02 February 2009.

[24] BASNEY, J., HUMPHREY, M., AND WELCH, V. The MyProxy Online Credential Repository. *Software: Practice and Experience 35*, 9 (June 2005), 801–816. DOI:10.1002/spe.688.

[25] BAYLISS, C. Use of AFS in the nanoCMOS Project. In *AFS & Kerberos Best Practices Workshop* (June 2009). `http://workshop.openafs.org/afsbpw09/talks/fri_1/AFS-within-the-nanoCMOS-project.pdf`, retrieved on 20 August 2012.

[26] BECK, A. High Throughput Computing: An Interview With Miron Livny. *HPCwire 6*, 25 (June 1997), 11444.

[27] BERMAN, F. *The Grid: Blueprint for a New Computing Infrastructure*, second ed. Morgan Kaufmann, 2004, ch. High-Performance Schedulers, pp. 279–309. ISBN 978-1-55860-933-4.

[28] BERNERS-LEE, T., HALL, W., HENDLER, J., SHADBOLT, N., AND WEITZNER, D. J. Computer Science: Enhanced: Creating a Science of the Web. *Science 313*, 5788 (August 2006), 769–771. DOI:10.1126/science.1126902.

[29] BERNSTEIN, K., FRANK, D. J., GATTIKER, A. E., HAENSCH, W., JI, B. L., NASSIF, S. R., NOWAK, E. J., PEARSON, D. J., AND ROHRER, N. J. High-performance CMOS variability in the 65-nm regime and beyond. In *Proceedings of the IEEE International Electron Devices Meeting 2007 (IEDM 2007)* (December 2007), pp. 569–571. DOI:10.1109/IEDM.2007.4419002, ISBN 978-1-4244-1507-6 (Print), 978-1-4244-1508-3 (Online).

[30] BRACCO, G., GIAMMARINO, L., MIGLIORI, S., SCIÓ, C., SANTORO, A., AND ROCCHI, A. AFS Pool Account Users – GSSKLOG and LCMAPS Extension to Support AFS Users as EGEE Pool Account Users. Tech. Rep. EGEE-TR-2006-006, CERN, August 2006. `http://cdsweb.cern.ch/record/990369/files/egee-tr-2006-006.pdf`, retrieved on 05 January 2009.

[31] BRADLEY, D., DASU, S., LIVNY, M., MOHAPATRA, A., TANNENBAUM, T., AND THAIN, G. *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09)*, vol. 219 of *Journal of Physics: Conference Series*. IOP Publishing, 2010, ch. Condor Enhancements for a Rapid-response Adaptive Computing Environment for LHC. DOI:10.1088/1742-6596/219/6/062035.

[32] BROADFOOT, P. J., AND MARTIN, A. P. A Critical Survey of Grid Security Requirements and Technologies. Tech. Rep. PRG-RR-03-15, Oxford University Computing Laboratory, August 2003. http://www.cs.ox.ac.uk/techreports/oucl/rr-03-15.html, retrieved on 19 February 2007.

[33] BURK, D. L. Intellectual Property in the Context of e-Science. *Journal of Computer-Mediated Communication 12*, 2 (January 2007). http://jcmc.indiana.edu/vol12/issue2/burk.html, retrieved on 13 April 2007.

[34] BUYYA, R. *GridSim 5.0 Beta API Specification*. Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne, Australia, 2009. http://www.buyya.com/gridsim/doc/api/overview-summary.html, retrieved on 11 August 2011.

[35] BUYYA, R., ABRAMSON, D., AND GIDDY, J. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In *The Fourth International Conference / Exhibition on High Performance Computing in the Asia-Pacific Region 2000* (May 2000), vol. 1, IEEE Computer Society, pp. 283–289. DOI:10.1109/HPC.2000.846563.

[36] BUYYA, R., ABRAMSON, D., GIDDY, J., AND STOCKINGER, H. Economic Models for Resource Management and Scheduling in Grid Computing. *Concurrency and Computation: Practice & Experience 14*, 13–15 (November / December 2002), 1507–1542. DOI:10.1002/cpe.690.

[37] BUYYA, R., AND MURSHED, M. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *Concurrency and Computation: Practice & Experience 14*, 13–15 (November / December 2002), 1175–1220. DOI:10.1002/cpe.710.

[38] CAFARO, M., AND ALOISIO, G. *Computer Communications and Networks*. Springer London, 2011, ch. Grids, Clouds and Virtualization, pp. 1–18. DOI:10.1007/978-0-85729-049-6_1, ISBN 978-0-85729-048-9 (Print) 978-0-85729-049-6 (Online), ISSN 1617-7975.

[39] CANTOR, S., KEMP, J., PHILPOTT, R., AND MALER, E. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Tech. rep., OASIS, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, retrieved on 23 January 2009.

[40] CARMODY, S. Shibboleth® Overview and Requirements. Tech. rep., Computing and Information Services, Brown University, February 2001.

[41] CASAVANT, T. L., AND KUHL, J. G. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. *IEEE Transactions on Software Engineering 14*, 2 (February 1988), 141–154. DOI:10.1109/32.4634, ISSN 0098-5589.

[42] CHADWICK, D., AND OTENKO, S. A Comparison of the Akenti and PERMIS Authorization Infrastructures in Ensuring Security in IT Infrastructures. In *Proceedings of the ITI First International Conference on Information and Communications Technology (ICICT 2003)* (November / December 2003), M. T. El-Hadidi, Ed., pp. 5–26.

[43] CHADWICK, D. W., AND OTENKO, A. The PERMIS X.509 Role Based Privilege Management Infrastructure. *Future Generation Computer Systems 19*, 2 (February 2003), 277–289. DOI:10.1016/S0167-739X(02)00153-X.

[44] CHADWICK, D. W., OTENKO, A., AND BALL, E. Role-Based Access Control With X.509 Attribute Certificates. *IEEE Internet Computing 7*, 2 (April 2003), 62–69. DOI:10.1109/MIC.2003.1189190, ISSN 1089-7801.

[45] CHAPIN, S. J., KATRAMATOS, D., KARPOVICH, J., AND GRIMSHAW, A. S. The Legion Resource Management System. In *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing* (1999), D. G. Feitelson and L. Rudolph, Eds., vol. 1659 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 162–178. DOI:10.1007/3-540-47954-6_9, ISBN 978-3-540-66676-9 (Print), 978-3-540-47954-3 (Online).

[46] CHAUDHRY, S., CAPRIOLI, P., YIP, S., AND TREMBLAY, M. High-Performance Throughput Computing. *IEEE Micro 25*, 3 (May / June 2005), 32–45. DOI:10.1109/MM.2005.49, ISSN 0272-1732.

[47] CHEN, A. Network Based Storage: Getting Students to Actually Use It. In *Proceedings of the 34th annual ACM SIGUCCS Conference on User Services* (New York, NY, USA, November 2006), ACM, ACM, pp. 55–59. DOI:10.1145/1181216.1181229, ISBN 1-59593-438-3.

[48] CHENG, B. J., ROY, S., ROY, G., AND ASENOV, A. Integrating 'Atomistic', Intrinsic Parameter Fluctuations into Compact Model Circuit Analysis. In *Proceedings of the 33rd Conference on European Solid-State Device Research 2003*

(ESSDERC '03) (September 2003), J. Franca and P. Freitas, Eds., pp. 437–440. DOI:10.1109/ESSDERC.2003.1256907, ISBN 0-7803-7999-3.

[49] CONDOR TEAM, UNIVERSITY OF WISCONSIN-MADISON. *Condor® Version 7.9.0 Manual*. `http://research.cs.wisc.edu/condor/manual/v7.9/index.html`, retrieved on 17 September 2012.

[50] CZAJKOWSKIY, K., FITZGERALDZ, S., FOSTERX, I., AND KESSELMAN, C. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing* (August 2001), IEEE Computer Society, pp. 181–194. DOI:10.1109/HPDC.2001.945188, ISBN 0-7695-1296-8, ISSN 1082-8907.

[51] DALHEIMER, M., AND PFREUNDT, F.-J. GenLM: License Management for Grid and Cloud Computing Environments. In *Proceedings of the 2009 9th IEEE / ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)* (May 2009), pp. 132–139. DOI:10.1109/CCGRID.2009.31, ISBN 978-1-4244-3935-5 (Print), 978-0-7695-3622-4 (Online).

[52] DAVENHALL, A. C., HARBULOT, B., JONES, M., STEWART, G., SINNOTT, R. O., ASENOV, A., MILLAR, C., ROY, G., AND REID, D. Data Management of Nanometre-Scale CMOS Device Simulations. Tech. rep., August 2009. `http://eprints.gla.ac.uk/7460/1/7460.pdf`, retrieved on 19 September 2012.

[53] DAVENHALL, A. C., HARBULOT, B., AND STEWART, G. OpenAFS Quick Start Guide: Installing and Using an OpenAFS Client. Tech. Rep. Revision 0.9b, National e-Science Centre / The University of Manchester, January 2009.

[54] DEFANTI, T. A., FOSTER, I., PAPKA, M. E., STEVENS, R., AND KUHFUSS, T. Overview of the I-Way: Wide-Area Visual Supercomputing. *International Journal of Supercomputer Applications and High Performance Computing 10*, 2 / 3 (Summer / Fall 1996), 123–131. DOI:10.1177/109434209601000201.

[55] DHAMIJA, R., AND WALLENBERG, F. A Framework for Evaluating Digital Rights Management Proposals. In *Proceedings of the First International Mobile IPR Workshop: Rights Management of Information Products on the Mobile Internet* (August 2003), Helsinki Institute for Information Technology (HIIT).

[56] DILLON, T., WU, C., AND CHANG, E. Cloud Computing: Issues and Challenges. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications* (April 2010), pp. 27–33. DOI:10.1109/AINA.2010.187, ISBN 978-1-4244-6695-5, ISSN 1550-445X.

[57] DIMITRAKOS, T. *Grid and Cloud Computing: A Business Perspective on Technology and Applications*. Springer Berlin Heidelberg, November 2010, ch. Common Capabilities for Service Oriented Infrastructures - Grid and Cloud Computing, pp. 123–145. DOI:10.1007/978-3-642-05193-7_8, ISBN 978-3-642-05192-0 (Print), 978-3-642-05193-7 (Online).

[58] DONG, F. A Taxonomy of Task Scheduling Algorithms in the Grid. *Parallel Processing Letters 17*, 4 (December 2007), 439–454. DOI:10.1142/S0129626407003149, ISSN 0129-6264 (Print), 1793-642X (Online).

[59] DONG, F., AND AKL, S. G. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Tech. Rep. 2006-504, School of Computing, Queen's University, Kingston, Ontario, January 2006. `http://ftp.qucis.queensu.ca/TechReports/Reports/2006-504.pdf`, retrieved on 03 February 2009.

[60] DOWNES, P., YAIKHOM, G., GIDDY, J., WALKER, D. W., SPEZI, E., AND LEWIS, D. High-Performance Computing for Monte Carlo Radiotherapy Calculations. *Philosophical Transactions of the Royal Society A 367*, 1897 (June 2009), 2607–2617. DOI:10.1098/rsta.2009.0028.

[61] DRESCHER, M., ANJOMSHOAA, A., WILLIAMS, G., AND MEREDITH, D. JSDL Parameter Sweep Job Extension. Community Practice GFD.149, Open Grid Forum, March 2009. `https://forge.ogf.org/sf/sfmain/do/go/artf6282`, retrieved on 05 February 2009.

[62] ELLISON, C. SPKI Requirements. RFC 2692, IETF, September 1999. `http://www.ietf.org/rfc/rfc2692.txt`, retrieved on 15 January 2009.

[63] ELLISON, C., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., AND YLONEN, T. SPKI Certificate Theory. RFC 2693, IETF, September 1999. `http://www.ietf.org/rfc/rfc2693.txt`, retrieved on 15 January 2009.

[64] ERNEMANN, C., KROGMANN, M., LEPPING, J., AND YAHYAPOUR, R. *Job Scheduling Strategies for Parallel Processing*, vol. 3277 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 2005, ch. Scheduling on the

Top 50 Machines, pp. 11–16. DOI:10.1007/11407522_2, ISBN 978-3-540-25330-3.

[65] FARINA, F., LACAPRARA, S., BACCHI, W., CINQUILLI, M., CODISPOTI, G., CORVO, M., DORIGO, A., FANFANI, A., FANZAGO, F., GUTSCHE, O., KAVKA, C., MERLO, M., SERVOLI, L., AND SPIGA, D. Status and Evolution of CRAB. In *Proceedings of the 11th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2007)* (Amsterdam, Netherlands, April 2007).

[66] FARRELL, S., AND HOUSLEY, R. An Internet Attribute Certificate Profile for Authorization. RFC 3281, IETF, April 2002. `http://www.ietf.org/rfc/rfc3281.txt`, retrieved on 23 January 2009.

[67] FERGUSON, A. OpenAFS and the Dawn of a New Era. In *AFS & Kerberos Best Practices Workshop* (May 2008). `http://workshop.openafs.org/afsbpw08/talks/wed_1/OpenAFS_and_the_Dawn_of_a_New_Era.pdf`, retrieved on 05 September 2012.

[68] FERRIS, C., AND FARRELL, J. What Are Web Services? *Communications of the ACM 46*, 6 (June 2003), 31. DOI:10.1145/777313.777335.

[69] FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[70] FIELDING, R. T., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999. `http://www.ietf.org/rfc/rfc2616.txt`, retrieved on 14 August 2012.

[71] FLECHL, M., AND FIELD, L. Grid Interoperability: Joining Grid Information Systems. *Journal of Physics: Conference Series 119*, 6 (2008). DOI:10.1088/1742-6596/119/6/062030, ISSN 1742-6596.

[72] FOSTER, I. What Is The Grid? A Three Point Checklist. *Grid Today 1*, 6 (July 2002).

[73] FOSTER, I. WS-Resource Framework: Globus Alliance Perspectives. Presented at globusworld 2004, The Globus Alliance, January 2004. `http://www.globus.org/wsrf/foster_wsrf.pdf`, retrieved on 02 February 2009.

[74] FOSTER, I., CZAJKOWSKI, K., FERGUSON, D. F., FREY, J., GRAHAM, S., MAGUIRE, T., SNELLING, D., AND TUECKE, S. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. *Proceedings of the IEEE 93*, 3 (March 2005), 604–612. DOI:10.1109/JPROC.2004.842766, ISSN 0018-9219.

[75] FOSTER, I., AND KESSELMAN, C. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications and High Performance Computing 11*, 2 (Summer 1997), 115–128. DOI:10.1177/109434209701100205.

[76] FOSTER, I., KESSELMAN, C., NICK, J. M., AND TUECKE, S. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Series in Communications Networking & Distributed Systems. John Wiley & Sons, Ltd., May 2002, ch. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, pp. 217–250. DOI:10.1002/0470867167.ch8, ISBN 978-0-4708-5319-1 (Print), 978-0-4708-6716-7 (Online).

[77] FOSTER, I., KESSELMAN, C., TSUDIK, G., AND TUECKE, S. A Security Architecture for Computational Grids. In *Proceedings of the 5th ACM conference on Computer and communications security* (New York, NY, USA, November 1998), SIGSAC: ACM Special Interest Group on Security, Audit and Control, ACM Press, pp. 83–92. DOI:10.1145/288090.288111, ISBN 1-58113-007-4.

[78] FOSTER, I., KESSELMAN, C., AND TUECKE, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications 15*, 3 (Fall 2001), 200–222. DOI:10.1177/109434200101500302.

[79] FOSTER, I., KESSELMAN, C., AND TUECKE, S. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004, ch. The Open Grid Services Architecture, pp. 215–257. ISBN 978-1-5586-0933-4.

[80] FOSTER, I., ZHAO, Y., RAICU, I., AND LU, S. Cloud Computing and Grid Computing 360-Degree Compared. In *Proceedings of the Grid Computing Environments Workshop (GCE '08)* (November 2008), pp. 1–10. DOI:10.1109/GCE.2008.4738445, ISBN: 978-1-4244-2860-1.

[81] FREY, J., TANNENBAUM, T., LIVNY, M., FOSTER, I., AND TUECKE, S. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Com-*

puting 5, 3 (July 2002), 237–246. DOI:10.1023/A:1015617019423, ISSN 1386-7857.

[82] GALLARDO, A., DE CERIO, L. D., MESSEGUER, R., ISERN-DEYÀ, A. P., AND SANJEEVAN, K. *Proceedings of the 9th International Conference on Computational Science (ICCS 2009), Part I*, vol. 5544/2009 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, 2009, ch. GRID Resource Searching on the GridSim Simulator, pp. 357–366. DOI:10.1007/978-3-642-01970-8_35, ISBN 978-3-642-01969-2.

[83] GARFINKEL, S. *PGP*. O'Reilly Media, Inc., 1995. ISBN 978-1-56592-098-9.

[84] GARFINKEL, S. *Architects of the Information Society: 35 Years of the Laboratory for Computer Science at MIT*. MIT Press, 1999. ISBN: 978-0-262-07196-3.

[85] GEELAN, J. Twenty-One Experts Define Cloud Computing. *Cloud Computing Journal 2009* (January 2009), 1–5.

[86] THE GLOBUS ALLIANCE. *GT 2.4: The Globus Resource Specification Language RSL v1.0*. `http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html`, retrieved on 17 June 2010.

[87] THE GLOBUS ALLIANCE. *Installing GT 5.2.0*. `http://www.globus.org/toolkit/docs/5.2/5.2.0/admin/install`, retrieved on 12 May 2012.

[88] THE GLOBUS ALLIANCE. *GT4 Admin Guide*, November 2005. `http://www.globus.org/toolkit/docs/4.0/admin/docbook/admin.pdf`, retrieved on 26 March 2010.

[89] GOODALE, T., JHA, S., KAISER, H., KIELMANN, T., KLEIJER, P., LASZEWSKI, G. V., LEE, C., MERZKY, A., RAJIC, H., AND SHALF, J. SAGA: A Simple API for Grid Applications. High-level Application Programming on the Grid. In *Computational Methods in Science and Technology* (2006), vol. 12.

[90] GOSLING, J., JOY, B., STEELE, G., AND BRACHA, G. *The Java™ Language Specification*, third ed. Addison-Wesley, May 2005. ISBN 0-321-24678-0.

[91] GOWERS, A. *Gowers Review of Intellectual Property*. The Stationery Office, December 2006. ISBN 978-011840483-9.

[92] GUDGIN, M., HADLEY, M., MENDELSOHN, N., MOREAU, J.-J., NIELSEN, H. F., KARMARKAR, A., AND LAFON, Y. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation, W3C, 27 April 2007. http://www.w3.org/TR/2007/REC-soap12-part1-20070427, retrieved on 02 February 2009.

[93] GUDGIN, M., HADLEY, M., AND ROGERS, T. Web Services Addressing 1.0 – Core. W3C Recommendation, W3C, 9 May 2006. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509, retrieved on 02 February 2009.

[94] HAAS, H. Reconciling Web Services and REST Services. In *Proceedings of the Third European Conference on Web Services (ECOWS 2005)* (2005). http://www.w3.org/2005/Talks/1115-hh-k-ecows, retrieved on 14 August 2012.

[95] HADLEY, M. Web Application Description Language. W3C Member Submission, W3C, 31 August 2009. http://www.w3.org/Submission/wadl, retrieved on 02 February 2009.

[96] HAGHIGHI, M. S., ZAHEDI, M. H., GHAZIZADEH, A. M., AND AHANGARY, F. *Innovations in Computing Sciences and Software Engineering*. Springer Netherlands, 2010, ch. Extending OpenMP for Agent Based DSM on GRID , pp. 265–271. DOI:10.1007/978-90-481-9112-3_45, ISBN 978-90-481-9112-3.

[97] HAN, L., ASENOV, A., BERRY, D., MILLAR, C., ROY, G., ROY, S., SINNOTT, R., AND STEWART, G. Towards a Grid-Enabled Simulation Framework for Nano-CMOS Electronics. In *Proceedings of IEEE e-Science 2007* (December 2007). DOI:10.1109/E-SCIENCE.2007.78, ISBN 978-0-7695-3064-2.

[98] HANDSCHUH, S., STAAB, S., AND MAEDCHE, A. CREAM – Creating Relational Metadata with a Component-based, Ontology-driven Annotation Framework. In *Proceedings of the 1st International Conference on Knowledge Capture* (October 2001), SIGART: ACM Special Interest Group on Artificial Intelligence, ACM, pp. 76–83. DOI:10.1145/500737.500752, ISBN 1-58113-380-4.

[99] HEIDL, S. Evaluierung von AFS/OpenAFS als Clusterdateisystem. Tech. rep., Zuse-Institut Berlin, July 2001. http://www2.cs.upb.de/StaffWeb/jens/Courses/VLZIB/Datamngmnt/ausarbeitung_sebastian_afs.pdf, retrieved on 18 December 2008.

[100] HEY, T., AND TREFETHEN, A. e-Science and its Implications. *Philosophical Transactions of the Royal Society A 361*, 1809 (August 2003), 1809–1825. DOI:10.1098/rsta.2003.1224.

[101] HOMER. *Iliad*. c. 9th – 8th Century BC, ch. Book XXII: The Death of Hector.

[102] HOSCHEK, W., JAN-MARTNEZ, F. J., SAMAR, A., STOCKINGER, H., AND STOCKINGER, K. Data Management in an International Data Grid Project. In *Proceedings of the First IEEE / ACM International Workshop on Grid Computing (Grid '00)* (2000), R. Buyya and M. Baker, Eds., vol. 1971 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 77–90. DOI:10.1007/3-540-44444-0_8, ISBN 978-3-540-41403-2 (Print), 978-3-540-44444-2 (Online).

[103] HOWARD, J. H. An Overview of the Andrew File System. In *Proceedings of the USENIX 1988 Winter Technical Conference* (1988), pp. 23–26.

[104] HOWE, T. *Crux for GT Developers*. The Globus Alliance, August 2009. `http://confluence.globus.org/display/whi/Crux+for+GT+Developers`, retrieved on 12 May 2012.

[105] HUEDO, E., MONTERO, R. S., AND LLORENTE, I. M. The GridWay Framework for Adaptive Scheduling and Execution on Grids. *Scientific International Journal for Parallel and Distributed Computing 6*, 3 (September 2005), 1–8. ISSN 1895-1767.

[106] HUMPHREY, M., SMITH, C., THEIMER, M., AND WASSON, G. JSDL HPC Profile Application Extension, Version 1.0. Tech. Rep. GFD-R-P.111, Open Grid Forum, August 2007. `http://www.gridforum.org/documents/GFD.111.pdf`, retrieved on 02 February 2010.

[107] HUMPHREY, M., AND THOMPSON, M. R. Security Implications of Typical Grid Computing Usage Scenarios. *Cluster Computing 5*, 3 (July 2002), 257–264. DOI:10.1023/A:1015621120332.

[108] HUNT, R. PKI and Digital Certificate Infrastructure. In *Proceedings of the Ninth IEEE International Conference on Networks* (October 2001), IEEE, pp. 234–239. DOI:10.1109/ICON.2001.962346, ISBN 0-7695-1187-4.

[109] IBM CORP. *IBM Tivoli Workload Scheduler Version 8.6: Scheduling Workload Dynamically*, 2011. SC23-9856-02. `http://pic.dhe.ibm.com/`

infocenter/tivihelp/v47r1/topic/com.ibm.tivoli.itws.doc_8.6.0.1/awsbrmst.pdf, retrieved on 19 September 2012.

[110] IN, J., AVERY, P., CAVANAUGH, R., CHITNIS, L., KULKARNI, M., AND RANKA, S. SPHINX: A Fault-Tolerant System for Scheduling in Dynamic Grid Environments. In *Proceedings of the 19th IEEE International Parallel and Distributed Computing Symposium* (April 2005), IEEE Computer Society, p. 12b. DOI:10.1109/IPDPS.2005.409, ISBN 0-7695-2312-9.

[111] JACOB, J., RAJSINGH, E. B., AND JESUDASAN, I. B. *Trends in Computer Science, Engineering and Information Technology*, vol. 204 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2011, ch. Dynamic Multi Dimensional Matchmaking Model for Resource Allocation in Grid Environment, pp. 179–185. DOI:10.1007/978-3-642-24043-0_19, ISBN 978-3-642-24043-0.

[112] JENSEN, J. G. UK e-Science Certification Authority Certificate Policy and Certification Practices Statement. Tech. Rep. Version 1.4, Science and Technology Facilities Council Rutherford Appleton Laboratory, November 2007. http://www.ngs.ac.uk/sites/default/files/cps-1_4.pdf, retrieved on 22 January 2009.

[113] JONES, M. A. S. *Using AFS as a Distributed File System for Computational and Data Grids in High Energy Physics*. PhD thesis, University of Manchester, 2005.

[114] JUSSIEN, N., PRUDHOMME, C., CAMBAZARD, H., ROCHART, G., LABURTHE, F., ET AL. choco: an Open Source Java Constraint Programming Library. In *Proceedings of the CPAIOR '08 Workshop on Open-Source Software for Integer and Constraint Programming (OSSICP '08)* (May 2010). http://hal.archives-ouvertes.fr/docs/00/48/30/90/PDF/choco-presentation.pdf, retrieved on 06 March 2012.

[115] KAHNG, A. B., LACH, J., MANGIONE-SMITH, W. H., MANTIK, S., MARKOV, I. L., POTKONJAK, M., TUCKER, P., WANG, H., AND WOLFE, G. Watermarking Techniques for Intellectual Property Protection. In *Proceedings of the 35th Annual Design Automation Conference (DAC '98)* (June 1998), vol. 35, ACM, pp. 776–781. DOI:10.1145/277044.277240, ISBN 0-89791-964-5.

[116] KARONIS, N. T., TOONEN, B., AND FOSTER, I. MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing 63*, 5 (May 2003), 551–563. DOI:10.1016/S0743-7315(03)00002-9, ISSN 0743-7315.

[117] KASHYAP, V., AND SHETH, A. *Cooperative Information Systems: Trends and Directions*. Academic Press, November 1997, ch. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies, pp. 139–178. ISBN 978-0-1254-4910-6.

[118] KEATING, M. *Devolution in Practice*. Institute for Public Policy Research, 2002, ch. Devolution and Public Policy in the United Kingdom: Divergence or Convergence?, pp. 3–23. ISBN 978-1-86030-199-5.

[119] KERTÉSZ, A., AND KACSUK, P. *Grid Middleware and Services: Challenges and Solutions*. CoreGRID series. Springer US, 2008, ch. Meta-Broker for Future Generation Grids: A New Approach for a High-Level Interoperable Resource Management, pp. 53–63. DOI:10.1007/978-0-387-78446-5_4, ISBN 978-0-387-78445-8 (Print), 978-0-387-78446-5 (Online).

[120] KOHL, J., AND NEUMAN, C. The Kerberos Network Authentication Service (V5). RFC 1510, IETF, September 1993. `http://www.ietf.org/rfc/rfc1510.txt`, retrieved on 05 January 2009.

[121] KOHNFELDER, L. M. Towards a Practical Public-key Cryptosystem. BS thesis, Massachusetts Institute of Technology, May 1978.

[122] KOSAR, T., AND BALMAN, M. A New Paradigm: Data-aware Scheduling in Grid Computing. *Future Generation Computer Systems 25*, 4 (April 2009), 406–413. DOI:10.1016/j.future.2008.09.006.

[123] KRAUTER, K., BUYYA, R., AND MAHESWARAN, M. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Software: Practice and Experience 32*, 2 (February 2002), 135–164. DOI:10.1002/spe.432.

[124] KREGER, H. Fulfilling the Web Services Promise. *Communications of the ACM 46*, 6 (June 2003), 29–34. DOI:10.1145/777313.777334.

[125] KUMAR, S., KUMAR, N., AND KUMAR, P. Genetic Algorithm for Network-Aware Job Scheduling in Grid Environment. In *Proceedings of Recent Advances*

in Intelligent Computational Systems (RAICS) (September 2011), IEEE, pp. 615–620. DOI:10.1109/RAICS.2011.6069384, ISBN 978-1-4244-9477-4.

[126] LANCASTER UNIVERSITY. *What is the HEC?* `http://www.lancs.ac.uk/iss/hec/what_is_hec.html`, retrieved on 18 June 2012.

[127] LEE, W., MCGOUGH, A. S., AND DARLINGTON, J. Performance Evaluation of the GridSAM Job Submission and Monitoring System. In *Proceedings of the UK e-Science All Hands Meeting* (September 2005), S. J. Cox and D. W. Walker, Eds. ISBN 1-904425-53-4.

[128] LEGRAND, A. Models, Simulation, Emulation, Experimentation for Grid Computing. In *Grid 5000 School* (March 2006). `http://mescal.imag.fr/membres/arnaud.legrand/articles/slides_g5k_simul.pdf`, retrieved on 04 August 2011.

[129] LI, J., WÄLDRICH, O., AND ZIEGLER, W. *From Grids to Service and Pervasive Computing*. Springer US, 2008, ch. Towards SLA-Based Software Licenses And License Management in Grid Computing, pp. 139–152. DOI:10.1007/978-0-387-09455-7_10, ISBN 978-0-387-09454-0 (Print), 978-0-387-09455-7 (Online).

[130] LIRA, M., DEVEREUX, C., TOOKE, D., AND CHURCHILL, J. The NGS at RAL. Tech. rep., Science & Technology Facilities Council, 2011. Retrieved on 18 June 2012.

[131] LITZKOW, M. J., LIVNY, M., AND MUTKA, M. W. Condor – A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems* (June 1988), pp. 104–111. DOI:10.1109/DCS.1988.12507, ISBN 0-8186-0865-X.

[132] LIU, B. J., ZHOU, M., AND DOCUMET, J. Utilizing Data Grid Architecture for the Backup and Recovery of Clinical Image Data. *Computerized Medical Imaging and Graphics 29*, 2–3 (March / April 2005), 95–102. DOI:10.1016/j.compmedimag.2004.09.004.

[133] LORCH, M., COWLES, B., GOMMANS, R. B. L., MADSEN, P., MCNAB, A., RAMAKRISHNAN, L., SANKAR, K., SKOW, D., AND THOMPSON, M. R. Conceptual Grid Authorization Framework and Classification. Memorandum GFD-

I.038, Global Grid Forum, November 2004. http://www.gridforum.org/documents/GFD.38.pdf, retrieved on 20 February 2007.

[134] MALCOLM, D. The Five Pillars of Cloud Computing. *Cloud Computing Journal 2009* (June 2009).

[135] MANDAL, P., THAKRAL, A., AND VERMA, S. Watermark Based Digital Rights Management. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)* (April 2005), vol. 1, pp. 74–78. DOI:10.1109/ITCC.2005.294, ISBN 0-7695-2315-3.

[136] MARCO, C., FABIO, C., ALVISE, D., ANTONIA, G., ALESSIO, G., FRANCESCO, G., ALESSANDRO, M., ELISABETTA, M., SALVATORE, M., AND LUCA, P. *Advances in Grid and Pervasive Computing*, vol. 5529/2009 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, April 2009, ch. The gLite Workload Management System, pp. 256–268. DOI:10.1007/978-3-642-01671-4_24, ISBN 978-3-642-01670-7.

[137] MARTIN, M., AGNEW, G., KUHLMAN, D. L., MCNAIR, J. H., RHODES, W. A., AND TIPTON, R. Federated Digital Rights Management – A Proposed DRM Solution for Research and Education. *D-Lib Magazine 8*, 7/8 (2002). DOI:10.1045/july2002-martin.

[138] MCCARTHY, J. LISP: A Programming System for Symbolic Manipulations. In *Preprints of papers presented at the 14th National Meeting of the Association of Computing Machinery* (New York, NY, USA, 1959), ACM, pp. 1–4. DOI:10.1145/612201.612243.

[139] MCNAB, R., AND HOWELL, F. W. Using Java for Discrete Event Simulation. In *Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop (UKPEW)* (September 1996), pp. 219–228.

[140] MESBAH, M., SARVI, M., TAN, J., AND KARIMIRAD, F. *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science*, vol. 145/2012 of *Advances in Intelligent and Soft Computing*. Springer Berlin / Heidelberg, 2012, ch. High Throughput Computing Application to Transport Modeling, pp. 45–51. DOI:10.1007/978-3-642-28308-6_7, ISBN 978-3-642-28307-9.

[141] METZ, C. AAA Protocols: Authentication, Authorization, and Accounting for the Internet. *IEEE Internet Computing Magazine 3*, 6 (November / December 1999), 75–79. DOI:10.1109/4236.807015, ISSN 1089-7801.

[142] MICELI, C., MICELI, M., JHA, S., KAISER, H., AND MERZKY, A. Programming Abstractions for Data Intensive Computing on Clouds and Grids. In *Proceedings of the 2009 9th IEEE / ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)* (May 2009), IEEE Computer Society, pp. 478–483. DOI:10.1109/CCGRID.2009.87, ISBN 978-0-7695-3622-4.

[143] MOORE, G. E. Cramming More Components onto Integrated Circuits. *Electronics 38*, 8 (April 1965), 114–117. DOI:10.1109/JPROC.1998.658762.

[144] NAGEL, L. W., AND PEDERSON, D. O. SPICE (Simulation Program with Integrated Circuit Emphasis). Tech. Rep. UCB/ERL M382, Electronics Research Laboratory, University of California, Berkeley, April 1973. Presented at the 16th Midwest Symposium on Circuit Theory, Waterloo, Ontario, Canada. `http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/ERL-382.pdf`, retrieved on 23 January 2009.

[145] THE NATIONAL GRID SERVICE. *UI-WMS Tutorial 2*. `http://wiki.ngs.ac.uk/index.php?title=UI-WMS_Tutorial2`, retrieved on 28 July 2011.

[146] THE NATIONAL GRID SERVICE. *Glasgow Computing Service*, November 2009. `http://www.ngs.ac.uk/glasgow/computing-service`, retrieved on 18 June 2012.

[147] THE NATIONAL GRID SERVICE. *Manchester NGS*, November 2009. `http://www.ngs.ac.uk/manchester/manchester-ngs`, retrieved on 18 June 2012.

[148] THE NATIONAL GRID SERVICE. *NGS CARMEN NGSII Cluster*, November 2009. `http://www.ngs.ac.uk/westminster`, retrieved on 18 June 2012.

[149] NATRAJAN, A., GRIMSHAW, A. S., HUMPHREY, M. A., AND NGUYEN-TUONG, A. Dispelling Seven Myths about Grid Resource Management. Tech. Rep. A770754, Department of Computer Science, University of Virginia, 2004. `http://www.anandnatrajan.com/papers/TechRep-CS-2004-33.pdf`, retrieved on 03 February 2009.

[150] NEEDLEMAN, M. Standards Update: The Shibboleth Authentication/Authorization System. *Serials Review 34*, 3 (August 2004), 252–253. DOI:10.1016/j.serrev.2004.05.006.

[151] NEOCLEOUS, K., DIKAIAKOS, M. D., FRAGOPOULOU, P., AND MARKATOS, E. P. Failure Management in Grids: The Case of the EGEE Infrastructure. *Parallel Processing Letters 17*, 4 (January 2007), 391–410. ISSN 0129-6264.

[152] NOVOTNY, J. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Series in Communications Networking & Distributed Systems. John Wiley & Sons, Ltd., May 2002, ch. The Grid Portal Development Kit, pp. 657–674. DOI:10.1002/0470867167.ch27, ISBN 978-0-4708-5319-1 (Print), 978-0-4708-6716-7 (Online).

[153] NOVOTNY, J., TUECKE, S., AND WELCH, V. An Online Credential Repository for the Grid: MyProxy. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing* (August 2001), IEEE Computer Society, pp. 104–111.

[154] NURMI, D., MANDAL, A., BREVIK, J., KOELBEL, C., WOLSKI, R., AND KENNEDY, K. Evaluation of a Workflow Scheduler Using Integrated Performance Modelling and Batch Queue Wait Time Prediction. In *Proceedings of the 2006 ACM / IEEE conference on Supercomputing* (2006), vol. November, ACM. DOI:10.1145/1188455.1188579, ISBN 0-7695-2700-0.

[155] OMII-UK. *GridSAM Client Module: JSDL Support*, October 2008. `http://gridsam.sourceforge.net/2.1.0/gridsam-client/jsdl.html`, retrieved on 20 August 2012.

[156] ORACLE CORPORATION. *Oracle® Grid Engine User Guide Release 6.2 Update 7*, February 2012. Part Number E21976-02. `http://docs.oracle.com/cd/E24901_01/doc.62/e21976.pdf`, retrieved 06 September 2012.

[157] OTTENS, B., DIMITRAKAKIS, C., AND FALTINGS, B. DUCT: An Upper Confidence Bound Approach to Distributed Constraint Optimization Problems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (July 2012), pp. 528–534.

[158] OXFORD E-RESEARCH CENTRE. *National Grid Service (NGS)*. `http://www.oerc.ox.ac.uk/downloads/posters/NGSPoster.pdf`, retrieved on 18 June 2012.

[159] PACINI, F., AND MARASCHINI, A. Job Description Language Attributes Specification for the gLite Middleware (Submission through WMProxy Service). Tech. Rep. EGEE-JRA1-TEC-590869-JDL-Attributes-v0-9, EGEE, November 2007.

[160] PAGANI, M. *Information Security and Ethics*. Idea Group Inc. (IGI), 2004, ch. The Critical Role of Digital Rights Management Processes in the Context of the Digital Media Management Value Chain, pp. 289–305. ISBN 978-1-5914-0286-2.

[161] PAPAZOGLOU, M. P., AND GEORGAKOPOULOS, D. G. Service-Oriented Computing. *Communications of the ACM 46*, 10 (October 2003), 24–28. DOI:10.1145/944217.944233, ISSN 0001-0782.

[162] PASKIN, N. *Encyclopedia of Library and Information Sciences*, third ed. Taylor & Francis, London, 2010, ch. Digital Object Identifier (DOI®) System, pp. 1586–1592. ISBN 0-8493-9712-X (Print), 0-8493-9711-1 (Online).

[163] PEARLMAN, L., KESSELMAN, C., WELCH, V., FOSTER, I., AND TUECKE, S. The Community Authorization Service: Status and Future. In *Proceedings of the 2003 Conference on Computing in High Energy and Nuclear Physics* (March 2003).

[164] PEARLMAN, L., WELCH, V., FOSTER, I., KESSELMAN, C., AND TUECKE, S. A Community Authorization Service for Group Collaboration. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks* (June 2002), pp. 50–59. DOI:10.1109/POLICY.2002.1011293, ISBN 0-7695-1611-4.

[165] PETERSON, H. E., AND TURN, R. System Implications of Information Privacy. In *Proceedings of the 1967 Spring Joint Computer Conference* (Santa Monica, CA, USA, April 1967), no. AD0650847, ACM, pp. 291–300. DOI:10.1145/1465482.1465526.

[166] PHAM, Q., DEVILLE, Y., AND HENTENRYCK, P. V. *Constraints*. Lecture Notes in Computer Science. Springer Netherlands, July 2012, ch. LS(Graph): A Constraint-based Local Search for Constraint Optimization on Trees and Paths, pp. 292–306. DOI:10.1007/s10601-012-9124-0, ISSN 1383-7133.

[167] PINEDO, M. L. *Scheduling: Theory, Algorithms, and Systems*, third ed. Springer, 2008. DOI:10.1007/978-0-387-78935-4, ISBN 978-0-387-78934-7 (Print), 978-0-387-78935-4 (Online).

[168] RAICU, I., FOSTER, I. T., AND ZHAO, Y. Many-Task Computing for Grids and Supercomputers. In *Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS 2008)* (November 2008), pp. 1–11. DOI:10.1109/MTAGS.2008.4777912, ISBN 978-1-4244-2872-4.

[169] REID, D., MILLAR, C., ROY, G., ROY, S., SINNOTT, R., STEWART, G., STEWART, G., AND ASENOV, A. Prediction of Random Dopant Induced Threshold Voltage Fluctuations in NanoCMOS Transistors. In *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices 2008 (SISPAD '08)* (September 2008), pp. 21–24. DOI:10.1109/SISPAD.2008.4648227, ISBN 978-1-4244-1753-7.

[170] REID, D., MILLAR, C., ROY, S., ROY, G., SINNOTT, R., STEWART, G., STEWART, G., AND ASENOV, A. Enabling Cutting Edge Semiconductor Simulation through Grid Technology. *Philosophical Transactions of the Royal Society A 367*, 1897 (June 2009), 2573–2584. DOI:10.1098/rsta.2009.0031.

[171] ROY, G., GHETTI, A., BENVENUTI, A., ERLEBACH, A., AND ASENOV, A. Comparative Simulation Study of the Different Sources of Statistical Variability in Contemporary Floating-Gate Nonvolatile Memory. *IEEE Transactions on Electron Devices 58*, 12 (December 2011), 4155–4163. DOI:10.1109/TED.2011.2167511, ISSN 0018-9383.

[172] SAGA, K., AIDA, K., AND MIURA, K. Mutual Job Submission Architecture That Considered Workload Balance Among Computing Resources in the Grid Interoperation. In *Proceedings of the 2011 IEEE / ACM 12th International Conference on Grid Computing (GRID '11)* (Washington, DC, USA, September 2011), IEEE Computer Society, pp. 19–25. DOI:10.1109/Grid.2011.12, ISBN 978-0-7695-4572-1.

[173] SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. Role-Based Access Control Models. *Computer 29*, 2 (February 1996), 38–47. DOI:10.1109/2.485845, ISSN 0018-9162.

[174] SANDHU, R. S., AND SAMARATI, P. Access Control: Principles and Practice. *IEEE Communications Magazine 32*, 9 (September 1994), 40–48. DOI:10.1109/35.312842, ISSN 0163-6804.

[175] SANJAY, H. A., AND VADHIYAR, S. Performance modeling of parallel applications for grid scheduling. *Journal of Parallel and Distributed Computing 68*, 8 (August 2008), 1135–1145. DOI:10.1016/j.jpdc.2008.02.006, ISSN 0743-7315.

[176] SAVVA, A. JSDL SPMD Application Extension, Version 1.0. Tech. Rep. GFD-R-P.115, Open Grid Forum, August 2007. `http://www.gridforum.org/documents/GFD.115.pdf`, retrieved on 19 September 2012.

[177] SCAVO, T., CANTOR, S., AND DORS, N. Shibboleth Architecture Technical Overview. Working Draft 02, Internet2 Middleware Initiative, June 2005. `http://open-systems.ufl.edu/files/draft-mace-shibboleth-tech-overview-latest.pdf`, retrieved 22 August 2012.

[178] SEE, S. Sun's Grid Model. In *Proceedings of the 1st IEEE International Workshop on Grid Economics and Business Models (GECON 2004)* (April 2004), pp. 173–184. DOI:10.1109/GECON.2004.1317596, ISBN 0-7803-8525-X.

[179] SILVA, P. F., WESTPHALL, C. B., WESTPHALL, C. M., AND ASSUNÇÃO, M. D. Composition of a DIDS by Integrating Heterogeneous IDSs on Grids. In *Proceedings of the 4th International Workshop on Middleware for Grid Computing* (New York, NY, 2006), ACM Press, pp. 12–17. DOI:10.1145/1186675.1186688, ISBN1-59593-581-9.

[180] SINNOTT, R., ASENOV, A., BERRY, D., BROWN, A., HAN, L., MILLAR, C., ROY, G., ROY, S., AND STEWART, G. Grid Infrastructures for the Electronics Domain: Requirements and Early Prototypes from an EPSRC Pilot Project. In *Proceedings of the UK e-Science All Hands Meeting* (September 2007), S. J. Cox, Ed. ISBN 978-0-9553988-3-4.

[181] SINNOTT, R., BAYLISS, C., DOHERTY, T., MARTIN, D., MILLAR, C., STEWART, G., AND WATT, J. Integrating Security Solutions to Support nanoCMOS Electronics Research. In *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications 2008 (ISPA '08)* (December 2008), pp. 71–79. DOI:10.1109/ISPA.2008.132, ISBN 978-0-7695-3471-8.

[182] SINNOTT, R. O., BAYLISS, C., DAVENHALL, C., HARBULOT, B., MILLAR, C., ROY, G., ROY, S., STEWART, G., WATT, J., AND ASENOV, A. Secure, Performance-Oriented Data Management for nanoCMOS Electronics. In *Proceedings of the IEEE 4th International Conference on e-Science 2008* (December 2008), pp. 87–94. DOI:10.1109/eScience.2008.21, ISBN 978-0-7695-3535-7.

[183] SINNOTT, R. O., CHADWICK, D. W., DOHERTY, T., MARTIN, D., NASSER, B., STELL, A., STEWART, G., SU, L., AND WATT, J. Advanced Security for Virtual Organizations: The Pros and Cons of Centralized vs. Decentralized Security Models. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid 2008 (CCGrid 2008)* (May 2008), pp. 106–113. DOI:10.1109/CCGRID.2008.67, ISBN 978-0-7695-3156-4.

[184] SINNOTT, R. O., DOHERTY, T., MARTIN, D., MILLAR, C., STEWART, G., AND WATT, J. *Proceedings of the 8th International Conference on Computational Science (ICCS '08), Part 1*, vol. 5101/2008 of *Lecture Notes in Computer Science*. Springer Verlag, June 2008, ch. Supporting Security-Oriented, Collaborative nanoCMOS Electronics Research, pp. 96–105. DOI:10.1007/978-3-540-69384-0_15, ISBN 978-3-540-69383-3.

[185] SINNOTT, R. O., STELL, A. J., CHADWICK, D. W., AND OTENKO, A. *Advances in Grid Computing – EGC 2005*, vol. 3470/2005 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, July 2005, ch. Experiences of Applying Advanced Grid Authorisation Infrastructures, pp. 265–274. DOI:10.1007/11508380_28, ISBN 978-3-540-26918-2.

[186] SINNOTT, R. O., STEWART, G., ASENOV, A., MILLAR, C., REID, D., ROY, G., ROY, S., DAVENHALL, C., HARBULOT, B., AND JONES, M. e-Infrastructure Support for NanoCMOS Device and Circuit Simulations. In *Proceedings of the Conference on Parallel and Distributed Computing and Networks* (February 2010), M. H. Hamza, Ed., ACTA Press.

[187] SMARR, L., AND CATLETT, C. E. Metacomputing. *Communications of the ACM 35*, 6 (June 1992), 44–52. DOI:10.1145/129888.129890, ISSN 0001-0782.

[188] SMITH, A., GERSTEIN, M., BERNERS-LEE, T., HALL, W., HENDLER, J., SHADBOLT, N., AND WEITZNER, D. J. Data Mining on the Web. *Science 314*, 5806 (December 2006), 1682. DOI:10.1126/science.314.5806.1682b.

[189] SOBIE, R., AGARWAL, A., ALLAN, J., BENNING, M., KOWALEWSKI, R., SMECHER, G., VANDERSTER, D., ZWIERS, I., HICKS, G. J., IMPEY, R., MATEESCU, G., AND QUESNEL, D. HEP Applications on the Grid Canada Testbed. In *Proceedings of the 2003 Conference on Computing in High Energy and Nuclear Physics* (March 2003).

[190] STEINER, J. G., NEUMAN, C., AND SCHILLER, J. I. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the USENIX 1988 Winter Technical Conference* (1988), pp. 191–202.

[191] STELL, A. J., SINNOTT, R. O., AND WATT, J. P. Comparison of Advanced Authorisation Infrastructures for Grid Computing. In *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS'05)* (May 2005), IEEE Computer Society, pp. 195–201. DOI:10.1109/HPCS.2005.20, ISBN 0-7695-2343-9.

[192] STEWART, G., MARKOV, S., ASENOV, P., MILLAR, C., REID, D., ROY, G., AND ASENOV, A. Automation of Variability-Aware CMOS Device and Circuit Design. Tech. rep., University of Glasgow, January 2011.

[193] STEWART, G., AND VANDERBAUWHEDE, W. Improving User Experience of Submitting Jobs to HPC Resources. In *Proceedings of the 2012 International Conference on High Performance Computing and Simulation* (July 2012), IEEE, pp. 635–641. DOI:10.1109/HPCSim.2012.6266985, ISBN 978-1-4673-2362-8.

[194] SUDLOW, R. OpenAFS: A HPC Filesystem? Presentation, University of Notre Dame, 2007. `http://crc.nd.edu/facilities/documents/afsbpw2007.pdf`, retrieved on 18 December 2008.

[195] SULISTIO, A., CIBEJ, U., VENUGOPAL, S., ROBIC, B., AND BUYYA, R. A Toolkit for Modelling and Simulating Data Grids: an Extension to GridSim. *Concurrency and Computation: Practice and Experience 20*, 13 (September 2008), 1591–1609. DOI:10.1002/cpe.1307.

[196] SUN MICROSYSTEMS, INC. *Sun N1 Grid Engine 6.1 User's Guide*. Santa Clara, CA, USA, May 2007. Part No.: 820-0699. `http://docs.oracle.com/cd/E19957-01/820-0699`, retrieved on 23 February 2010.

[197] TALIA, D. The Open Grid Services Architecture: Where the Grid Meets the Web. *IEEE Internet Computing 6*, 6 (November / December 2002), 67–71. DOI:10.1109/MIC.2002.1067739, ISSN 1089-7801.

[198] TANG, M., LEE, B.-S., TANG, X., AND YEO, C.-K. The impact of data replication on job scheduling performance in the Data Grid. *Future Generation Computer Systems 22*, 3 (February 2006), 254–268. DOI:10.1016/j.future.2005.08.004, ISSN 0167-739X.

[199] TEMPLETON, D. Better Preemption. Blog post, Oracle Corporation, 28 January 2010. `https://blogs.oracle.com/templedf/entry/better_preemption`, retrieved on 17 September 2012.

[200] THOMPSON, M., JOHNSTON, W., MUDUMBAI, S., HOO, G., AND JACKSON, K. Certificate-based Access Control for Widely Distributed Resources. In *Proceedings of the 8th USENIX Security Symposium* (August 1999).

[201] TOLONE, W., AHN, G.-J., PAI, T., AND HONG, S.-P. Access Control in Collaborative Systems. *ACM Computing Surveys 37*, 1 (2005), 29–41. DOI:10.1145/1057977.1057979, ISSN 0360-0300.

[202] TOMÁS, L., CAMINERO, A. C., RANA, O., CARRIÓN, C., AND CAMINERO, B. A GridWay-based Autonomic Network-aware Metascheduler. *Future Generation Computer Systems 28*, 7 (July 2012), 1058–1069. DOI:10.1016/j.future.2011.08.019, ISSN 0167-739X.

[203] TOWN, C., AND HARRISON, K. Large-Scale Grid Computing for Content-Based Image Retrieval. *Aslib Proceedings 62*, 4/5 (2010), 438–446. DOI:10.1108/00012531011074681, ISSN 0001-253X.

[204] TUECKE, S., CZAJKOWSKI, K., FOSTER, I., FREY, J., GRAHAM, S., KESSELMAN, C., MAGUIRE, T., SANDHOLM, T., SNELLING, D., AND VANDERBILT, P. Open Grid Services Infrastructure (OGSI). Specification Version 1.0, Global Grid Forum, June 2003. `http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf`, retrieved on 02 February 2009.

[205] TUECKE, S., WELCH, V., ENGERT, D., PEARLMAN, L., AND THOMPSON, M. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC

3820, IETF, June 2004. http://www.ietf.org/rfc/rfc3820.txt, retrieved on 22 January 2009.

[206] ULLMAN, J. D. Polynomial Complete Scheduling Problems. *ACM SIGOPS Operating Systems Review 7*, 4 (October 1973), 96–101. DOI:10.1145/957195.808055, ISSN 0163-5980.

[207] VÁZQUEZ-POLETTI, J. L., HUEDO, E., MONTERO, R. S., AND LLORENTE, I. M. A Comparison Between Two Grid Scheduling Philosophies: EGEE WMS and GridWay. *Multiagent and Grid Systems 3*, 4 (December 2007), 429–439. ISSN 1574-1702.

[208] VILJOEN, M., AND CHURCHILL, J. NGS Uniform Execution Environment. Tech. Rep. Version 0.5, STFC RAL, UK, February 2010. http://www.ngs.ac.uk/sites/default/files/file/docs/UniformExecutionEnvironment.pdf, retrieved on 17 June 2010.

[209] WALKER, J. A., SINNOTT, R., STEWART, G., HILDER, J. A., AND TYRRELL, A. M. Optimising Electronic Standard Cell Libraries for Variability Tolerance Through the Nano-CMOS Grid. *Philosophical Transactions of the Royal Society A 368*, 1925 (August 2010), 3967–3981. DOI:10.1098/rsta.2010.0150.

[210] WATT, J., AND SINNOTT, R. O. Supporting Federated Multi-authority Security Models. In *Proceedings of the International Symposium on Cluster, Cloud and Grid Computing (CCGrid)* (May 2011), pp. 620–621. DOI:10.1109/CCGrid.2011.77, ISBN 978-1-4577-0129-0 (Print), 978-0-7695-4395-6 (Online).

[211] WATT, J., SINNOTT, R. O., JIANG, J., DOHERTY, T., STELL, A. J., MARTIN, D., AND STEWART, G. Federated Authentication & Authorisation for e-Science. In *Proceedings of the APAC Conference and Exhibition 2007* (October 2007).

[212] WELCH, V., FOSTER, I., KESSELMAN, C., MULMO, O., PEARLMAN, L., GAWOR, J., MEDER, S., AND SIEBENLIST, F. X.509 Proxy Certificates for Dynamic Delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop* (April 2004), NIST.

[213] WELCH, V., SIEBENLIST, F., FOSTER, I., BRESNAHAN, J., CZAJKOWSKI, K., GAWOR, J., KESSELMAN, C., MEDER, S., PEARLMAN, L., AND TUECKE, S.

Security for Grid Services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC03)* (June 2003), pp. 48–57. DOI:10.1109/HPDC.2003.1210015, ISBN 0-7695-1965-2.

[214] WHITE, B. S., WALKER, M., HUMPHREY, M., AND GRIMSHAW, A. S. LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications. In *ACM / IEEE Conference on Supercomputing* (November 2001), pp. 19–28. DOI:10.1109/SC.2001.10024, ISBN 1-58113-293-X.

[215] WONG, S. C., TAN, V., FANG, W., MILES, S., AND MOREAU, L. Grimoires: Grid Registry with Metadata Oriented Interface: Robustness, Efficiency, Security. In *Proceedings of the Work in Progress Session, Cluster Computing and Grid (CCGrid '05)* (May 2005).

[216] WOOD, C., KITE, S., TAYLOR, S. J., AND MUSTAFEE, N. Developing a Grid Computing System for Commercial-Off-The-Shelf Simulation Packages. In *Proceedings of the Operational Research Society Simulation Workshop (SW10)* (March 2010), pp. 184–191.

[217] YEO, C. S., AND BUYYA, R. A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing. *Software: Practice and Experience 36*, 13 (November 2006), 1381–1419. DOI:10.1002/spe.725.

[218] YOO, A. B., JETTE, M. A., AND GRONDONA, M. *Job Scheduling Strategies for Parallel Processing*, vol. 2862. Springer-Verlag, Berlin Heidelberg, 2003, ch. SLURM: Simple Linux Utility for Resource Management, pp. 44–60. DOI:10.1007/10968987_3.

[219] YU, J., AND BUYYA, R. A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing 3*, 3 – 4 (September 2005), 171–200. DOI:10.1007/s10723-005-9010-8, ISSN 1570-7873, 1572-9814.

[220] ZAYAS, E. R. AFS-3 Programmers' Reference: Architectural Overview. Tech. Rep. FS-00-D160, Transarc Corporation, September 1991. Version 1.0 of 2 September 1991 22:53. `http://stuff.mit.edu/afs/sipb/project/openafs/cvs-checkout/openafs-stable/doc/pdf/archov-doc.pdf`, retrieved on 05 January 2009.