Risk Assessment Models for Resource Failure in Grid Computing

By

Raid Abdullah Alsoghayer

Submitted in accordance with the requirements

for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

February, 2011

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

© 2011 The University of Leeds and Raid Alsoghayer

Acknowledgements

This work would never been possible without the guidance of my supervisor Dr. Karim Djemame. His wealth of experience has provided me with encouragement and critical feedback and for that I thank you. I would also like to thank the Collaborative Systems and Performance Research Group at the School of Computing for much insightful discussion and enjoyable comradeship.

Finally my heartiest gratitude goes to my wife, daughter, father & mother. Without your constant encouragement, un-conditional love and support I would not been able to undertake this endeavour.

Abstract

Service Level Agreements (SLAs) are introduced in order to overcome the limitations associated with the best-effort approach in Grid computing, and to accordingly make Grid computing more attractive for commercial uses. However, commercial Grid providers are not keen to adopt SLAs since there is a risk of SLA violation as a result of resource failure, which will result in a penalty fee; therefore, the need to model the resources risk of failure is critical to Grid resource providers. Essentially, moving from the best-effort approach for accepting SLAs to a risk-aware approach assists the Grid resource provider to provide a high-level Quality of Service (QoS). Moreover, risk is an important factor in establishing the resource price and penalty fee in the case of resource failure.

In light of this, we propose a mathematical model to predict the risk of failure of a Grid resource using a discrete-time analytical model driven by reliability functions fitted to observed data. The model relies on the resource historical information so as to predict the probability of the resource failure (risk of failure) for a given time interval. The model was evaluated by comparing the predicted risk of failure with the observed risk of failure using availability data gathered from Grids resources.

The risk of failure is an important property of a Grid resource, especially when scheduling jobs optimally in relation to resources so as to achieve a business objective. However, in Grid computing, user-centric scheduling algorithms ignore the risk factor and mostly address the minimisation of the cost of the resource allocation, or the overall deadline by which the job must be executed completely. Therefore, we propose a novel user-centric scheduling algorithm for scheduling Bag of Tasks (BoT) applications. The algorithm, which aims to meet user requirements, takes into account the risk of failure, the cost of resources and the job deadline. With this in mind, through simulation, we demonstrate that the algorithm provides a near-optimal solution for minimizing the cost of executing BoT jobs. Also, we show that the execution time of the proposed algorithm is very low, and is therefore suitable for solving scheduling problems in real-time.

Risk assessment benefits the resource provider by providing methods to either support accepting or rejecting an SLA. Moreover, it will enable the resource provider to understand the capacity of the infrastructure and to thereby plan future investment. Scheduling algorithms will benefit the resource provider by providing methods to meet user requirements and the better utilisation of resources. The ability to adopt a risk assessment method and user-centric algorithms makes the exploitation of Grid systems more realistic.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

- **Probabilistic Risk Assessment for Resource Provision in Grids**. R. Alsoghayer and K. Djemame. Proceedings of the 25th UK Performance Engineering Workshop, Leeds, UK, July 2009, pp. 99-110.
- **Modeling the Risk of Failure in Grid Environments**. R. Alsoghayer and K. Djemame. Proceedings of the 2010 International Conference on Grid Computing and Applications (GCA'2010), Las Vegas, Nevada, July 2010

Contents

Figures

Tables

Abbreviations

AMPL	Modelling Language for Mathematical Programming
BoT	Bag of Tasks
CCS	Cluster Computing Software
Cdf	Cumulative Distribution Function
DCI	Distributed Computing Infrastructure
DRFC	Deadline and Risk of Failure Constraints algorithm
EGEE	Enabling Grids for E-sciencE
EGI	European Grid Infrastructure
ENISA	European Network and Information Security Agency
GOCDB	Grid Operations Centre Data Base
GRAAP-WG	Grid Resource Allocation and Agreement Protocol Working Group
GRAM	Grid Resource Allocation and Management
GT	Globus Toolkit
HPP	Homogeneous Poisson Process
LP	linear programming
LRM	Local Resource Manager
LSF	Load Sharing Facility
MDS	Monitoring and Discovery System
MIP	Mixed Integer Programming
MIPS	Million Instructions per Second
MLE	Maximum Likelihood Estimation
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
NBQS	Network Batch Queuing Systems
NGS	National Grid Service
NHPP	Non-Homogeneous Poisson Process
NWS	Network Weather Services
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
OGSI	Open Grid Service Infrastructure
P2P	Peer-to-Peer
PACE	Performance Analysis and Characterisation Environment

PBS	Portable Batch System
Pdf	Probability Density Function
QoS	Quality of Services
RBD	Reliability Block Diagram
RE	Risk Exposure
RMS	Resource Management System
ROCOF	Rate of Occurrence of Failures
ROF	Risk of Failure
SGE	Sun Grid Engine
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
UNICORE	Uniform Interface to Computing Resources
WS	Web Services
WS-Agreement	Web Services Agreement
WSDL	Web Service Description Language
WSIL	Web Service Introspection Language
WSLA	Web Service Level Agreement
WSRF	Web Service Resource Framework
XML	Extensible Mark-up Language

Chapter 1

Introduction

1.1 Research Motivation

Grid computing [1]—much like many other computing technologies before it, such as the internet or the web—was initially motivated by the needs of scientists. As a result, this has increased the opportunity for collaboration between educational and research institutions, and accordingly broadened access to expertise and services through the sharing of resources. The need to share resources in order to achieve common goals is not limited to science and is fundamental to commerce; whether to support business processes across partners in a supply chain or to otherwise enable higher utilisation of resources spread across business units, Grid technologies are becoming increasingly applied in a wide range of businesses and commercial activities [2]. Another major driving force for Grid computing, from a business perspective, is that users can concentrate on their business applications as opposed to having to maintain complex in-house computing infrastructures. This will remove the large investment overhead associated with developing in-house computing infrastructures, and thereby reduce the overall costs associated with running and maintaining the business. Finally, computing infrastructures do not have to be sized on peak load but can use Grid technologies to cleverly share the burden in peak hours. This will reduce the cost of developing, running and maintaining a computing infrastructure, without incurring any notable decrease in performance.

Even with the huge commercial benefits of Grid computing, the commercial uptake of Grid computing has been slow, with the current Grid middleware (e.g. Globus Toolkit [3]) still follows the best-effort approach. Importantly, Grid users do not get any assurances that their applications will complete according to their requirements. Furthermore, commercial Grid resource providers are not attracted either: for a resource provider, agreeing to execute a user application without

enough information regarding the state and availability of resources introduces the risk of not fulfilling user requirements, which consequently results in a penalty fee paid by the resource provider. Moreover, there is a hazard attached to resources failure, service unavailability, insufficient resources, etc., all of which could lead to users' requirements not being fulfilled. Without a method for assessing the risk of agreeing to execute a user application, providers are only able to make uncertain decisions regarding suitable users' requests.

Essentially, improving the overall Quality of Services (QoS) of the Grid infrastructure so as to overcome the best-effort approach is one of the most important on-going issues in the Grid community [4]. Furthermore, providing greater integration, efficiency and QoS encourages users and businesses to exploit Grid infrastructure for commercial benefits. With this in mind, QoS can be characterised into two categories: qualitative and quantitative. Qualitative attributes are hard to measure, and refer to user satisfactions, trust and the reliability of the Grid provider. Essentially, the quantitative attribute can be measured exactly, and include user requirements, such as network, CPU or storage. For example, 'the response time should be less than 10 milliseconds' or 'the free memory should be more than 2 Gigabytes'. A number of requirements should be delivered by the Grid resource provider when striving to provide QoS. According to [5], these requirements include advance resource reservation, reservation policy, agreement protocol, security, simplicity and scalability. Another approach of delivering QoS is through the use of fault tolerance mechanisms, such as the replication or redundancy of resources. Owing to the fact that the probability of resource failure is higher in Grid environments than in a traditional distributed system, fault tolerance can improve the offered QoS [6].

Grid resources failures are frequent, and have fatal effects in relation to job execution; even the use of fault tolerance approaches cannot completely eliminate the effect of failures. For a Grid resource provider, such failures are a threat to jobs running on the Grid, as an unexpected resource failure may lead to user requirements not being fulfilled, which subsequently results in a penalty fee. If a resource failure affects the overall execution of a time critical application, results can be delayed or lost entirely, which therefore has consequences in the real world and can ultimately lead to broken commitments. Notably, this can have a knock-on

effect in other walks of life and, as such, Grid resource failures impose a risk on both the Grid provider and the user.

The word ‘risk’ is used in a variety of different disciplines/contexts, and has a different meaning for each. Even in a single corporation, different departments have different definitions for the term. In the Oxford English Dictionary, for example, ‘risk’ is defined as *‘[Exposure to] the possibility of loss, injury, or other adverse or unwelcome circumstance; a chance or situation involving such a possibility’* [7]. As different entities of a single corporation have different definitions for the concept, they also have different views: for example, Health and Safety department personnel view risk as a bad thing or a negative force; thus, any risk to the health and safety of company employees or to the public is to be avoided, or the probability and consequence of that risk are to be reduced to the greatest extent possible. On the other hand, finance personnel might hold different views, owing to the fact that one aspect of their job is to conduct a risk/reward evaluation. In this regard, greater risk usually yields greater returns, and so they view risk as a positive force [8].

Generally, risk is associated with the uncertainty of a future event, or a hazard, which might have a potentially negative impact on an asset by depreciating some of its attribute value. The uncertainty can be modelled in terms of probability provided sufficient information is known. In this sense, the probability is only considered as the occurrence of an event without any consideration to the consequences or the impacts of such an event; therefore, the word ‘risk’ is used to combine the probability of events with the impact or the expected losses of those events.

In order to make sound business decisions—such as outsourcing computing infrastructure—users need assurance that the Grid provider is able to guarantee their requirements. Moreover, they need to assess the risk of an unsatisfactory outcome and to thereby compare different Grid providers. The Grid provider needs to provide guarantees to users based on the current infrastructure. Furthermore, the provider needs to understand the capacity of the infrastructure and plan future investment.

In order to address the risk of Grid resources failures, methods to identify the events causing failures are needed, as well as estimations regarding the probability or frequency, and measurement of the expected losses of those events. A risk assessment model for estimating the risk of Grid resource failures provides a

solution to the risk problem, and would increase the chances of Grid commercial take-up, as well as helping in building trust in the Grid.

Consider the following scenario. A user submits an application execution request, as a Service Level Agreement (SLA), to the Grid resource provider, which is to be executed in-line with QoS requirements, such as timely execution. The provider is then required to estimate the risk of failure for each available resource owing to the fact that the resource risk of failure significantly influences the price of the resource and the penalty fee; presumably, a resource with low risk of failure is more expensive than the one with high risk of failure. The provider will then allocate resources to the user's application based on the user requirements and the estimated risk of failure; this will help the provider to decide whether the user application should be accepted or rejected. Furthermore, it will inform the user of the rate of not getting the desired QoS, and accordingly provide the opportunity to select the desired level of realistic guarantees. The work presented in this thesis proposes a number of mechanisms for estimating the Grid resources risk of failure, which will meet the requirements of such a scenario.

1.2 Thesis Objectives

The aim of this research is to study resources failure in the context of Grid computing. This is to estimate the risk of a resource failure, allowing users to compare different Grid resources and providers, and to thereby select the resources with the most acceptable level of risk. Furthermore, Grid resource providers use the resources risk as measures to guarantee users requirements.

With this in mind, the objectives of this thesis are:

- To develop a mathematical model which estimates the Grid resource risk of failure. Notably, moving from the best-effort approach in managing resources to a risk-aware approach which assists the Grid provider in offering a high-quality service. This will increase provider's revenue and demand for resources, whilst decreasing penalty fees to be paid in the event of user requirements violation. Furthermore, the reputation of the provider will improve so that additional users become motivated to outsource part of their computing activities to the provider.

- To consider resources risk of failure in the development of a scheduling algorithm for minimising the cost of executing applications on the Grid whilst simultaneously ensuring the application owners' constraints are fulfilled.

1.3 Methodology

The aim of this thesis is twofold: to develop a mathematical model to estimate the Grid resource risk of failure and to develop a scheduling algorithm for minimising the cost of executing application on the Grid. The research methodology to achieve these objectives is described below (Figure 1).

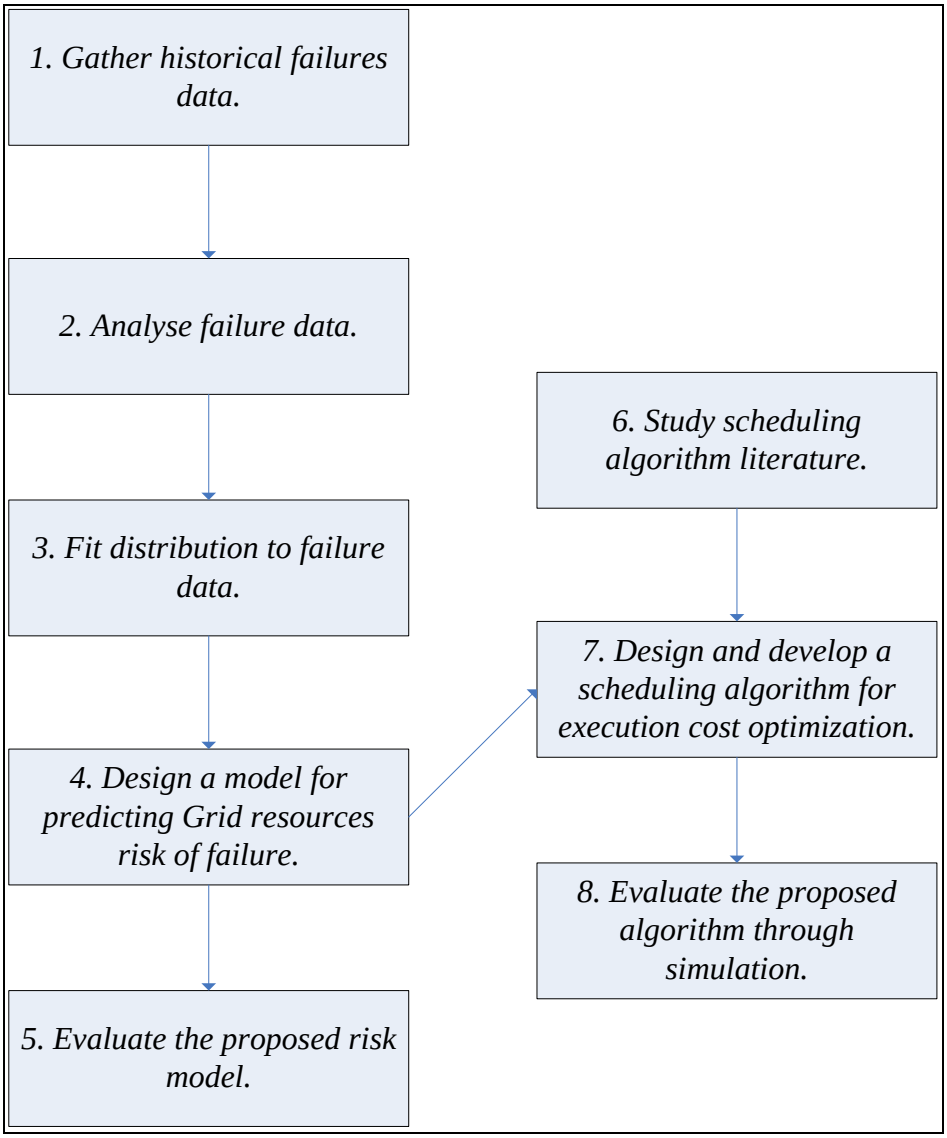


Figure 1: Research Methodology Diagram.

1. *Gather historical failures data*. The failure data is collected from seven Grid resources from two Grid sites. The behaviour of these data will be analysed to support the prediction of the Grid resources probability of failure.

2. *Analyse failure data*. The failure data is analysed with respect to three important properties of system failures: root cause, time to repair and time between failures.

3. *Fit distribution to failure data*. This is to interpret failure data for a variable in order to drive a distribution that realistically models its true variability. The empirical cumulative distribution function for the time between failures in the failure data is fitted with four standard distributions: Exponential, Weibull, Gamma and Lognormal.

4. *Design a model for predicting Grid resources risk of failure*. The Markov models techniques [177] are utilised to develop the risk of failure model.

5. *Evaluate the proposed risk model*. The collected failure data are used to compute the observed risk of failure and the risk model is evaluated by comparing the observed risk of failure with the risk of failure predicted by the proposed model. The two-sample t test is used to validate the comparison.

6. *Study scheduling algorithm literature*. This study is to identify the Grid users' requirements. These are heavily influenced by business objectives which rely on execution of Grid applications in a timely and cost-effective manner.

7. *Design and develop a scheduling algorithm for execution cost optimization*. The algorithm examines if including risk of failure as a scheduling requirement will improve current scheduling algorithms.

8. *Evaluate the proposed algorithm through simulation*. The purpose of the simulation is to test the allocation of Grid jobs with the use of the proposed scheduling algorithm, and to accordingly compare it with the optimal allocation of the jobs. Two criteria are used in the evaluation

the difference in the number of jobs allocated and the difference in the execution cost.

1.4 Major Contributions

The major contributions of this thesis include:

- A mathematical model to predict the Grid resources risk of failure. The model is also used to rank Grid resources, and shows the effects of future investments. The model is developed after detailed analysis of Grid resource failures using failure data collected from different Grid resources and spanning for three years. The analysis focuses on the statistical properties of the failure data, including the root cause of failures, the mean time between failures, and the mean time to repair.
- The development of an efficient algorithm—known as Deadline and Risk of Failure Constraints (DRFC) algorithm—. The algorithm determine a near-optimal execution cost for the mathematical model for minimising the costs associated with executing Bag of Tasks applications whilst ensuring the applications owners' constraints are fulfilled. The performance evaluation of the DRFC algorithm compared to the cost-minimising mathematical model optimal solution is conducted using simulation.

1.5 Thesis Overview

- Chapter 2 reviews background material, which helps to scope the area of research, followed by Grid resource management techniques and their challenges. A review of current scheduling approaches is provided. Finally, a survey of scheduling algorithms for the Bag of Tasks application is provided.
- Chapter 3 presents the risk theory and the various different approaches for risk assessment. A review of risk identification and assessment methods for Grid-based systems is presented, along with possible risk treatments.

- Chapter 4 begins by identifying the events causing resource failures. The method adapted to measuring the risk of resource failure is presented. Subsequently, before the analysis of Grid resource failures is presented, an overview of Grid resources failures data is provided along with the data-collection process. The models for Grid resources failures and repairs are provided.
- Chapter 5 presents the mathematical model to predict the risk of failure of a Grid resource using a discrete time analytical model driven by reliability functions fitted to observed failures data. The model evaluation and its uses to rank resources and direct future investments are showcased.
- Chapter 6 begins by providing a mathematical model for scheduling Bag of Tasks application on the Grid, and an efficient algorithm for solving the mathematical model is presented. The design of the simulation experiments is highlighted, along with the resource model and the workload model. Finally, the evaluation of the algorithm through simulation is discussed.
- Chapter 7 summarises the work on a chapter-by-chapter basis and outline some future work.

Chapter 2

Resource Allocation in Grid Systems

This chapter examines the definition and background of Grid computing in Section 2.1. Section 2.2 lists various types of application and Grid systems, as well outlining various Grid projects. Section 2.3 presents the Grid architecture. Section 2.4 provides a description of Grid middleware. Section 2.5 provides a description of Service Level Agreements and technologies which exist so as to facilitate their usage in Grid environments. Section 2.6 discusses Grid resource management technique and their challenges, and also comprises a survey of scheduling algorithms. Finally section 2.7 summaries the chapter.

2.1 Grid Computing

Computer scientists in the mid-1990s began to explore a new technology known as metacomputing. The interest was to link supercomputing sites [11]. The I-WAY project—which was introduced in the ACM/IEEE conference on Supercomputing 1995 (SC 95) and aims to unifying the resources of large US supercomputing sites—was the first step in the field [12]. The I-WAY project was essential to the understanding and progress of the emerging new technology [13]. The evolution from metacomputing through to Grid computing occurred with the introduction of middleware, which was designed in order to function as a wide-area infrastructure to support data-intensive applications and diverse online processing [14]. Currently, the Grid is defined as the coordinated sharing of resources and solving problems in dynamic, multi-institutional virtual organisations. This sharing must be controlled with clear boundaries regarding what will be shared, who are permitted to share, and the conditions under which sharing occurs, as well as whether the resources are hardware, software, or users [14, 15]. The sharing should also be carried out with the use of standard, open, and general-purpose protocols and interfaces, and should deliver nontrivial quality of services (QoS) [12, 15, 16].

2.2 Grid Applications

The interest in Grids is motivated by the novel uses of computers to solve complex applications. These applications provide the useful information and services for the reality of Grids.

2.2.1 Type of Applications

A survey of four general classes of applications that runs on Grid systems is given in [17]. It is summarised as follows:

- **Distributed supercomputing:** (also known as metacomputing): these systems solve very large and intensive problems with the use of multiple computers to achieve greater processing power. Many of the existing Grid systems and their applications are based on this class.
- **Real-time widely distributed instrumentation systems:** these systems involve real-time data sources. These systems rely on distributed-storage, network-based caches, agent-based monitoring, and generalized access control.
- **Data-intensive computing:** these systems are both data and compute intensive. These applications focus on processing and analysing information and require terabytes or petabytes of data to be processed and stored.
- **Teleimmersion:** these systems combine advance display technologies, computers, and networks to create shared virtual environments for collaborative design, education, and entertainments.

These are the general types of Grid application, and some applications may be of more than one type. A list of applications, their motivations for using Grid technologies and the architectures and approaches adopted in implementations are available in [18].

2.2.2 Types of Grid Systems

Notably, the types of Grid system are not identical; essentially, they vary widely in terms of both function and purpose. Krauter *et al.* [19] classify Grid systems into three categories:

- **Computational Grids [20]:** denotes systems which have higher total computational capacity available for single applications than the capacity of any constituent machine in the system. Computational Grids are amongst the first type of Grid systems. An important objective of Computational Grids is to benefit from the under-utilised computational resources through sharing.
- **Data Grid [21]:** denotes systems which provide an infrastructure for synthesising new information from data repositories, such as data warehouses or digital libraries, which are distributed in a wide area network. Many scientific applications require access to a large amount of data; therefore, data Grids are important when striving to increase the performance and to thereby achieve high throughput.
- **Service Grid [22]:** denotes systems which provide services that are not provided by any single machine. This category is further subdivided as on-demand, collaborative, and multimedia Grid Systems. An on-demand Grid category dynamically aggregates different resources so as to provide new services, e.g. allocating new machines to a simulation. A collaborative Grid connects users and applications into collaborative workgroups. A multimedia Grid provides an infrastructure for real-time multimedia applications.

Other types of Grid systems include.

- **eScience Grids:** denotes system devoted primarily to the solution of problems from science and engineering. Such Grids support the access to computational and data resources required in order to solve complex problem arising in the science communities. Enabling Grids for E-sciencE (EGEE) [23], Grid 5000 [24], and National Grid Service (NGS) [25] are some examples of e-Science Grids.
- **Enterprise Grids:** Grid computing is becoming an important component of business in the modern world. E-business must be able to adjust dynamically and efficiently to increases in demands or market shifts. The Grid offers a large potential to solving business problems by enabling

active projects within one large enterprise to share resources in a transparent way.

- **Desktop Grids:** this is the use of idle cycles of desktop Personal Computers (PCs). Desktop Grids are a new form of Enterprise Grids. Small enterprises are usually equipped with hundreds of desktops which can be utilised for setting up a Grid system.

2.2.3 Grid Projects

Grid technology is being used in many different areas of research and industry, and there are currently numerous projects utilising them. The following shortlist is to highlight the scale and diversity of projects being developed currently in Europe:

- **Enabling Grids for E-sciencE (EGEE) [23]:** is a project which brings together experts from more than 50 countries for developing a service Grid infrastructure, which is available to scientists 24 hours a day. The project encourages researchers in academia and business to collaborate and provide them with access to a production-level Grid infrastructure, independent of their geographic location. The EGEE infrastructure is the largest collaborative production Grid infrastructure in the world for e-science. Through EGEE, scientists are able to do more science and on a larger scale, and to therefore gather results in a shorter timeframe, which would not been possible without Grid technologies. EGEE closed in April, 2010, and the infrastructure is now supported by the new organisation EGI.eu [26], which is being developed to coordinate the European Grid Infrastructure (EGI).

- **Integrated Sustainable Pan-European Infrastructure for Researchers in Europe (EGI-InSPIRE) [27]:** this is a project to provide a sustainable and reliable European Grid Infrastructure (EGI) for European scientists and their international partners. Importantly, this integrates new Distributed Computing Infrastructures, such as clouds, supercomputing networks and desktop Grids. EGI-InSPIRE project focuses on application areas of high-energy physics, computational chemistry and life sciences.

- **A worldwide e-infrastructure for computational neuroscientists (outGRID) [28]:** this is a project concerned with igniting the process of converting the three active e-infrastructures for computational neuroscience

into one unique worldwide facility. The three e-infrastructures are neuGRID project [29], which provides large sets of brain images paired with grid-based computationally intensive algorithms for studies of neurodegenerative diseases, CBRAIN Project [30] and LONI [31], which offer computational resources and algorithm pipelines.

- **Desktop Grids for International Scientific Collaboration (DEGISCO) [32]:** this is a project which connects the European Distributed Computing Infrastructure (DCI) to International Cooperation Partner Countries (ICPC). The European DCI is already interconnected by EDGeS [33], and DEGISCO will further extend the infrastructure to ICPC countries. The DEGISCO project will support the creation of new Desktop Grids in ICPC countries, and the connection of these Grids to European DCIs.

Note that this list is by no means exhaustive. Grid projects the School of Computing at the University of Leeds has been involved with include Distributed Aircraft Maintenance Environment (DAME) [34], Business Resource Optimisation for Aftermarket & Design on Engineering Networks (BROADEN) [35], Advanced Risk Assessment & Management for Trustable Grids (AssessGrid) [36, 37], Integration Broker for Heterogeneous Information Sources (IBHIS) [38] and A Demand-Led Service-Based Architecture for Dependable e-Science Applications (e-Demand) [39].

2.3 Grid Architecture

2.3.1 Early Architecture

Grid architecture organizes components into layers. Components within each layer share common characteristics. Figure 2 is taken from [15], and describes a high level view of the Grid architecture. The architecture contains five layers and the following is a brief description of each one [1, 15, 16, 40].

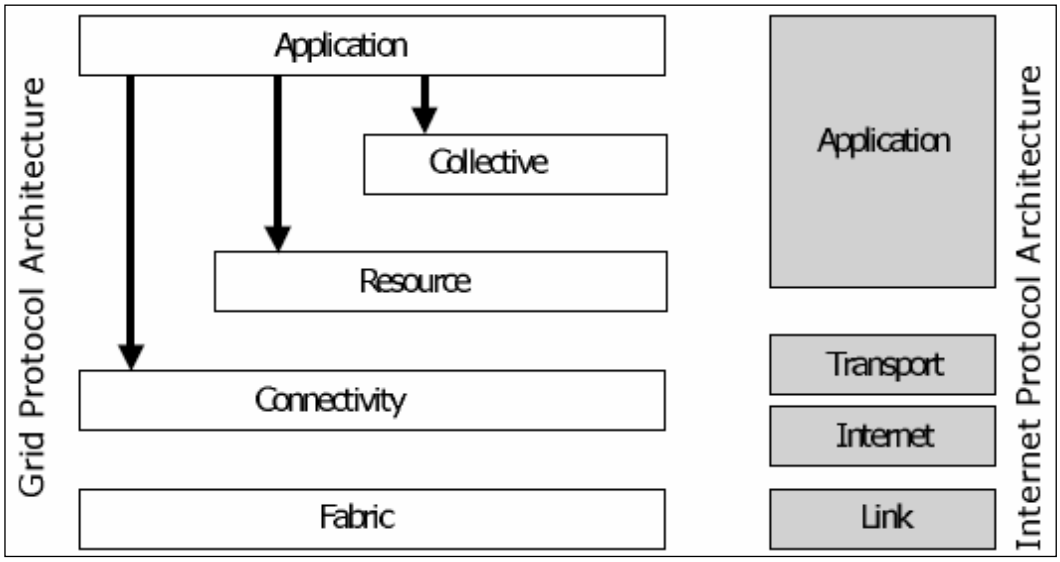


Figure 2: Layered Grid Architecture [15].

2.3.1.1 Grid Fabric Layer

The Grid fabric layer provides access to shared resources; these resources can be physical or logical. Notably, there is tight interdependence between functions implemented on the fabric and the supported sharing operations, which means richer fabric functionality enables sophisticated sharing operation. On the other hand, light fabric simplifies the development of the Grid. At a minimum, resources should implement introspection mechanisms that allow discovery of their structure, state, and capability, on the one hand, and resource management mechanisms that provide control of delivered QoS, on the other.

The shared resources can be divided into three main types of resources.

- **Computational Resources**, these are the physical machines that do the processing. Four types of computational resources are suggested in [17], and are summarised her.
    - **End user systems:** These are common computer machines; they have a single-functional entry and are homogeneous.
    - **Clusters:** These are a group of linked computers, working together closely and are most often highly homogeneous. Clusters are usually deployed in order to improve performance and/or availability over that of a single computer, whilst typically being much more cost-effective than single computers of comparable speed or availability.

- **Intranets:** These are large local networks of resources within a single organisation; they are diverse and heterogeneous by nature, and different parts of the network may be under different administration, which results in less global knowledge regarding the resources.
- **Extranets:** These are networks of Intranets. They span multiple organisations and are more heterogeneous than Intranets and have less global knowledge available.

- **Storage Resources**: These are dedicated storage machines which can hold very large amounts of data. This may be a simple file system or a large and complex database.
- **Network Resources**: These are the cable switches and routers that make the physical network. The network is measured by capacity (bandwidth) and latency.

2.3.1.2 Grid Connectivity Layer

This layer defines core communication and authentication protocols. The communication protocols are to enable the exchange of data between resources. Authentication protocols, which are built on the communication protocols, are required to provide secure mechanisms for checking users and resources. The most important requirements for security include:

- **Single sign-on:** the user should sign on only once and use as many remote resources as desired, if permitted, without the need to sign on to each resource.
- **Delegation:** the user must be able to give a program the ability to run on her/his behalf, so that the program can access resources the user has access to. Furthermore, the program itself could delegate a subset of its permission to a subprogram.
- **Integration with local security solution:** each resource on the Grid has it own security solutions, and provides different users with different solutions. A Grid security solution must interoperate with these solutions without the need for amendments.

- **User-based trust relation:** the user should be able to use different sites without requiring interaction between these sites.

2.3.1.3 Grid Resource Layer

This layer is built on the protocols of the connectivity layer, and defines protocols for secure negotiation, initiation, monitoring, control, accounting, and payment of sharing resources. The two primary protocols on this layer are information protocols, and management protocols.

2.3.1.4 Grid Collective Layer

This layer contains protocols and services not linked with a specific resource but instead containing interaction across collection of resources. This can enable the implementation of a wide variety of sharing behaviours without placing new requirements on the resources being shared.

2.3.1.5 Grid Application Layer

This layer contains the user applications. The applications are implemented by calling services defined at any layer.

2.3.2 Open Grid Services Architecture and Web Service Resource Framework

The Open Grid Forum (OGF) [41] —previously known as the Global Grid Forum (GGF)—is leading the global standardisation effort for grid computing. The OGF is a very large community of users, developers and vendors from industry and research, representing over 400 organisations in more than 50 countries.

Open Grid Services Architecture (OGSA) [42] was a key proposal from OGF. OGSA defines the architecture in terms of Grid Services, aligning it with Web Services (WS) technologies [43]. From the set of technologies in WS, the OGSA exploits the Simple Object Access Protocol (SOAP) [44], Web Service Description Language (WSDL) [45] and Web Service Introspection Language (WSIL) [46]. The OGSA underlying infrastructure—the Open Grid Service Infrastructure (OGSI) [47] —defines an extension on the use of WSDL so as to enable stateful Web services. It defines approaches for:

- creating, naming, and managing the lifetime of instances of services;
- declaring and inspecting service state data;

- the asynchronous notification of service state change;
- representing and managing collections of service instances; and
- common handling of service invocation faults.

OGSI has attracted criticism from the WS community, stating that the OGSI is too large and did not have separation (factoring) between functions to support incremental adoption. It also does not work well with existing WS and Extensible Mark-up Language (XML) [48] tooling. Furthermore, it is too object-oriented and encapsulates the resource state in the WS to model a resource [49].

The Web Service Resource Framework (WSRF) [50] was proposed in order to tackle the limitations of OGSI. It can be viewed as a straightforward refactoring of functionality within the OGSI in a manner that exploits development in WS technologies. The following are the components of WSRF specification: WS-Resource, WS-Addressing, WS-ResourceLifetime, WS-ResourceProprieties, WS-RenewableReferences, WS-ServiceGroup, WS-BaseFaults and WS-Notification.

2.4 Grid Middleware

Grid Middleware is a software layer that enables a seamless access to heterogeneous environments, such that the differences between platforms, network protocols, and administrative boundaries become completely transparent [51]. The main requirements for Grid middleware include:

- **Communication Services:** Grid applications' communication requirements are diverse, and the need to support network protocols and QoS parameters is essential. The communication services role is to provide such protocols.
- **Information Services:** A Grid is a dynamic enticement where the location and availability of Grid services changes rapidly. The monitoring and discovery of resources and services is vital for effectively utilising the resources. The information services enable the monitoring and discovery of resources and services.
- **Data Management:** Data in the Grid environment is stored in a distributed file system or distributed database. Data management services

responsibilities provide data replication and reliable file transfers so as to enable file redundancy, indexing and transfer between sites.

- **Security:** Enables the delegation of credential and authentication, which subsequently enables Grid users to invoke several Grid services from different sites without the need for authentication at each individual site (single sign-on).
- **Resource Allocation and Management:** Enables an efficient and effective application scheduling and execution on the Grid resources. Methods for locating, executing and terminating Grid services are provided. Furthermore, it is important for resource management services to have an interface with a local resource manager and a network batch queuing system so as to enable the local usage policies.

There has been a remarkable amount of effort in the design and implementation of middleware software for Grid computing. The following are two of the most successful and widely used middleware.

2.4.1 Globus Toolkit

The Globus Toolkit (GT) [3, 15, 16] has emerged as the *de facto* standard for Grid infrastructures. It was developed by the Argonne National Laboratory in the late 1990s with the objective to support the development of service-oriented distributed computing applications and infrastructures. Globus provides services and protocols to overcome the Grid problem. With this in mind, it is up to developers to deploy these services so as to support a range of different applications.

GT5 [52] is the most recent release, and has a set of service implementation, three containers to host the developer code, and a set of client libraries. The most important service is the Grid Resource Allocation and Management (GRAM), which provides a web interface for initiating, monitoring, and managing the execution of the application on the Grid [3, 15, 16, 53]. Other important services include data access and movement, e.g. Grid File Transfer Protocol (GridFTP) [54], Reliable File Transfer (RTF) [55], and Open Grid Service Architecture—Database Access and Integration (OGSA-DAI) [56], security and credential management, e.g. MyProxy, Delegation, and SimpleCA. The current version of GT is compliant with the OGSA and WSRF.

2.4.2 UNICORE

UNICORE [57, 58] —Uniform Interface to Computing Resources—is a Java-based middleware implementing a three-tier architecture comprising client, server and target system. The client tier supports the creation, manipulation and control of complex jobs, which can be executed on different sites running the UNICORE middleware; the server tier is the secure entry point into a UNICORE site, which is known as the Gateway, the role of which is to authenticate requests from the client tier and forward them to a Network Job Supervisor (NJS) for mapping into concrete jobs or actions which are performed by the target system; and the target system tier provides the Target System Interface (TSI), which resides on the host to interface with the local batch system on behalf of the user. In order to increase performance, multiple TSIs may be started on a single host.

2.4.3 Other Middleware

Other Middleware software applied in Grid systems includes gLite [59, 60], which was developed as part of the EDEE Middleware Re-engineering and Integration Research Activity. China Research and the development environment Over Wide-area Networks (CROWN) [61]. OMII-UK [62] —previously known as Open Middleware Infrastructure Institute (OMII)—is an open-source repository of Grid middleware components, services and tools.

2.5 Grid Service Level Agreements

Grid computing has relied on ‘best effort’ as the guiding principal of operation [63]. Although this approach is acceptable for non-commercial Grid environments, it is not the case for commercial Grid environments. Commercial Grid users require some form of commitment and assurance on top of the allocated resources, such as performance, security, availability, latency, etc., sometimes referred to as QoS. Commitment and assurances are specified in terms of Service Level Agreements (SLAs). SLAs either provide some measurable capability or perform a specific task, and thereby allow Grid users to know what is expected from a service without requiring detailed knowledge of the service providers’ policies [64, 65].

Service Level Agreements is outside the scope of this thesis, yet in Chapter 6 the focus is on the resource provider being able to schedule users’ application and

accordingly guarantee their requirements and constraints. In that chapter the users' requirements and constraints are assumed to be known, in the real world this is achieved through the use of SLAs. Therefore, the remainder of this section is dedicated to SLAs.

Importantly, there are various differences between commercial and non-commercial Grid users and providers. The distinction between the users and providers in non-commercial Grids is difficult because, most of the time, the group contributing resources to the Grid are also its users. On the other hand, however, commercial Grids have a strong differentiation between providers and users. Furthermore, commercial Grid users pay for the services, and so the expectations are high; with this in mind, users won't tolerate being denied service or being rescheduled to a different time slot.

Importantly, there have been a number of attempts to define SLAs architecture for Grid environments. Sahai *et al.* [63] propose a language for unambiguous and precise specification of SLAs, and a monitoring architecture for their evaluation. Moreover, Leff *et al.* [66] propose an architecture which utilises a dynamic offload mechanism so as to balance load on a commercial Grid provider's resources in order to efficiently meet SLAs requirements. Furthermore, Ludwig *et al.* [67] propose a novel SLA language for Web services. Standardising the way of establishing agreements between a recourse provider and a resource user is crucial for the wide adaption of SLAs. Accordingly, the following provides a description of two of the standardisation efforts.

2.5.1 Web Service Level Agreement (WSLA)

The Web Service Level Agreement (WSLA) [68] is an SLA language to support the specifying and monitoring of QoS guarantees within Web Services Framework. WSLA is based on XML, and comprises flexible and extendable XML Schema and a runtime architecture containing several SLA monitoring services. WSLA enables service users and service providers to unambiguously define an SLA, specify the SLA parameters and metrics, as well as the way in which the metrics are to be measured, and accordingly relate them to managed resource instrumentations. The elements of WSLA are Parties, Service Description, and Obligation. Notably, the parties section consists of a description of the parties involved in an SLA. The service description section specifies the characteristics of

the service and its observable parameters. Finally, the obligation section defines various guarantees and constraints that may be imposed on the SLA parameters.

2.5.2 WS-Agreement

The Grid Resource Allocation and Agreement Protocol Working Group (GRAAP-WG) [69] —which is a part of OGF—proposed the Web Services Agreement specification (WS-Agreement) [70] in order to establish an agreement between two parties using an extendable XML language. The specification includes three parts: a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement lifecycle. For compatibility and complexity, the WS-Agreement only defines the general structure of the agreement, which makes the implementation of WS-Agreement open; this allows the defining of domain-specific extensions or specific languages for expressing conditions [71]. Owing to the fact that the implementation of WS-Agreement is open, the Creation and Monitoring of Agreements (Cremona) [72] provides a layered agreement management architecture, which defines mechanisms to implement WS-Agreement interactions and connects them to the service provider system and the user system. It also implements the WS-Agreement interfaces, and provides management functionality for both the agreement templates and instances.

2.6 Resource Management

Grid resources are distributed on the globe with different administrative domains and geographic locations. In order for the Grid to provide coordinate-access to resources, regardless of the heterogeneous nature of the resources or their geographic locations, a number of challenging problems must be countered [53], as listed below:

- **Site autonomy:** refers to the fact that resources are owned and operated by different organisations, in different administrative domains.
- **Heterogeneous substrate:** derived from the site autonomy problem, and refers to the fact that different sites can use different local resource management systems. Notably, even if the same resource management is

used on more than one site, a different configuration leads to different functionality.

- **Policy extensibility:** as Grid resources are drawn from a wide range of domains—each with its own requirements—a resource management system must support the regular development of new domain-specific management structures.
- **Co-allocation:** some Grid jobs have resource requirements which cannot be satisfied using a single site. These requirements might be satisfied using resources, simultaneously, at several different sites. Owing to site autonomy and the possibility of resource failure during allocation, there is the need for specialised mechanisms which are able to collect resources information and submit jobs to multiple resources to guarantee the jobs requirements.
- **Online control:** a type of negotiation might be required in order to adapt application requirements to resource availability—especially when requirements and resource characteristics change during run time.

Resource management systems for distributed computing can be divided into two classes [53]:

- **Network batch queuing systems (NBQS):** These systems focus on resource management issues for computers in a network; they do not address policy extensibility or provide limited support for online control and co-allocation; and
- **Wide-area scheduling systems:** Resource management on these systems is performed through mapping application components to resources and scheduling their execution. These systems do not support heterogeneous substrates, site autonomy or co-allocation.

NBQS handles jobs by allocating resources from a networked pool of computers. Some examples of these systems include Load-Sharing Facility (LSF) [73], Portable Batch System (PBS) [74], and LoadLeveler [75]. The user of these systems characterises the requirements of the job to run either explicitly through a kind of job control language or implicitly through selecting which queue to submit the job to. Network batch queuing systems are designed for single administrative

domains, therefore making the site autonomy difficult to achieve. Furthermore, these systems assume they are the only resource management system in use, consequently further complicating the heterogeneous substrate problem. Policy extensibility is limited in these systems, and the end user has little control regarding how his/her requirement is interpreted.

Wide-area scheduling systems are usually distributed over several sites, and are more adoptable than NBQS system. Two popular wide-area scheduling systems are Legion [76, 77], which become Avaki commercial product, and Condor [78, 79].

Grid resource management systems do not have full control over resources. Resources exist in different administrative domains; they are heterogeneous and operate under different policies. As a result, the aforementioned systems, whilst addressing some of the difficulties in Grid resource management, do not cover all the issues [14]. The Grid Resource Allocation and Management (GRAM) implemented within Globus Toolkit provides a basic solution to the resource management problem. Moreover, GRAM resides on top of the local resource manager systems (LRM), and consists of several different components, which work together to authenticate users, manage jobs, interface with the LRM, and stage files. These components are described below.

- **Gatekeeper:** the gatekeeper service is responsible for the authentication and authorisation of the user's request, and also for starting up the job manager service. One instance of this daemon is created for each job submission.
- **Job Manager:** the job manager service is responsible for processing job requests and coordinating file transfers. One long-lived instance of this daemon is created for each LRM and one short-lived instance for each job.
- **Job Manager Script:** the job manager script process is responsible for interacting with LRM via the LRM adaptor.
- **Job Manager LRM Adaptor:** the LRM adaptor interacts directly with LRM, and is loaded into the job manager script component upon start-up.
- **Scheduler Event Generator:** the responsibility of the scheduler event generator process is parsing the LRM-specific data related to the job start-

up, execution, and termination into a general format independent from the LRM.

- **Scheduler Event Generator LRM Module:** the scheduler event generator LRM module process the LRM state to produce the event which the scheduler event generator writes into event log.
- **GRAM Audit Database:** the job manager can be configured to write audit into files, and the GRAM audit database program loads these file into a database.

Three stages are required in the process of Grid resource management, namely resource discovery, resource scheduling, and job execution and monitoring [80]. The Grid resource management system must be able to first discover available resources. Subsequently, it will select candidate resources for the job to be executed on. This selection is depending on the job requirements and the information gathered by the resource manager. Finally, the job is submitted to the local resource manager for execution and monitoring [81]. A taxonomy of Grid Resource Management Systems (RMS) can be found in [19].

In this thesis the focus is on scheduling (chapter 6), yet scheduling hugely depends on information gathered by the other stages of resource management, namely resource discovery and monitoring. Therefore, section 2.6.1 presents an overview of resource discovery. Section 2.6.2 provides a detailed overview of Grid scheduling. The 2.6.2 section is further divided into three subsections, type of scheduling in Grid systems, predicting execution time and scheduling algorithms. The algorithm developed in this thesis (Section 6.4) assumes the knowledge of execution time. Therefore, methods for predicting execution time are showcased. Also a survey of scheduling algorithms and their limitation is provided. Section 2.6.3 presents monitoring as a requirement for job scheduling within Grid systems.

2.6.1 Resource Discovery

The discovery of Grid resources is a very challenging problem owing to the diversity, large number, and dynamic behaviour of resources in the Grid. Grid information services [82] provide a mechanism for the discovery of distributed resources. The Monitoring and Discovery System (MDS) [83]—which is a part of the Globus Toolkit—is a classic example of information services. Notably, MDS has

undergone several major changes since it was first introduced. The latest MDS release is a suite of web services concerned with monitoring and discovering resources and services on Grids. It has two WSRF-based services: an index service and a trigger service. The former collects information from various sources and publishes that data as resource properties. In this instance, Grid users utilise the standard WSRF resource property query and subscription/notification interfaces in order to retrieve the resources information, so as to aid them in selecting suitable resources. Moreover, resource property entries in the index have a limited life-span, and will be removed if it is not renewed again before it expires. The design of indexes facilitates a hierarchical model, and thereby enables indexes to register with each other to aggregate data at several levels.

Information services mainly use centralised or static hierarchical models to discover resources. Other research works considered decentralised service discovery mechanisms—especially peer-to-peer (P2P) techniques [84, 85]. The benefits associated with using P2P systems include load-balancing, self-organisation, adaptation, and fault-tolerance, although P2P systems also have their own limitations. Essentially, they offer limited data management facilities, usually focus only on a single functionality, and offer different levels of reliability for individual peers.

2.6.2 Scheduling

Scheduling is assigning appropriate resources to incoming jobs. The assignment of resources can be carried out in a blind way; however, it is more profitable to use more advance scheduling technique. Thus, a Grid scheduler must automatically and efficiently find the most appropriate assignment of resources.

The scheduling problem is not limited to Grid systems. In fact, it is one of the most studied problems in the operation and optimisation research communities. However, in the case of Grid systems, the scheduling problem is different and more challenging. According to Xhafa & Abraham [86], the characteristics that make the Grid scheduling problem challenging are the following:

- **Dynamic structure of the Grid:** Resources in a Grid system cross different administrative domains, which makes the resources control very difficult. Furthermore, the resources join or leave the Grid system in an

unpredictable way; this could be owing to losing connection or the resources administrator switching off the resources, disconnecting the resources from the Grid system in order to carry out other important internal work, or even updating the resources operating system, etc.

- **High heterogeneity of resources:** In Grid systems, the resources are very heterogeneous and diverse, ranging from personal digital assistants PDAs, desktops, laptops, clusters, supercomputers and even special computational devices.
- **High heterogeneity of jobs:** Jobs arriving to any Grid system are heterogeneous, and could adopt computing-intensive or data-intensive application.
- **High heterogeneity of interconnecting networks:** Grid resources are connected together with different interconnection networks. Network performance (e.g. transmission speed, cost, latency, etc.) are all very important in the overall performance of Grid systems.
- **Existence of local scheduler:** Grid systems cross different administrative domains (e.g. universities, enterprises, research institutions, etc.), and most of these administrative domains have a local scheduler to run the Grid and local application. Therefore, a Grid scheduler must have the ability to interact with and accordingly use the available local schedulers.
- **Existence of local policies on resources:** Again, owing to the different administrative domains in the Grid, one cannot assume full control over the resources. Each administrative domain has it own set of policies that must be taken into account.
- **Jobs resource requirements:** Current schedulers assume full availability and compatibility of resources; however, this is not the case in real situations, as many restrictions and incompatibilities can arise from job and resource specifications.
- **The large scale of the Grid system:** One of the benefits of the Grid systems is the scalability and the ability to tackle large computational problems which cannot be solved using local resources. Therefore, Grid

schedulers are required to effectively manage resources in order to achieve scalability.

- **Security:** Security is fundamental in the case of Grid scheduling. This may refer to either the job or application security requirements, or the Grid resource security requirements. This characteristic is non-existing in classical scheduling.

2.6.2.1 Type of Scheduling in Grid Systems

Scheduling in Grid systems depends on two factors: the job requirements and the Grid environment characteristics. Different jobs could have different scheduling needs, such as batch or immediate scheduling. Furthermore, the Grid environments impose restrictions, such as the use of local scheduler. With this in mind, the following are the main types of scheduling in Grid environments and a scheduler might fit into more than one type.

- **Independent Scheduling:** Although much computer-science research has been carried out in direct relation to parallel processing, sequential jobs are still predominant in the real world of Grid Computing, and a large fraction of the jobs in the workloads imposed on such systems is owing to sequential applications—often submitted in the form of Bags of Tasks (BoT) [87]. The reasons behind this observable fact include the relatively high network latencies, the complexities of parallel programming models, and the nature of the computational work. BoT jobs are composed of sequential independent tasks where there is no communication or dependency amongst tasks. Examples of Bag of Tasks applications include Monte Carlo simulations, massive searches (such as key-breaking), image manipulation applications, data-mining algorithms, and parameter-sweep applications. Tasks in theses applications are scheduled independently.

- **Workflows Scheduling:** Solving many complex problems—especially e-Science applications—requires the combination and orchestration of Grid resources, such as computational devices, data, applications and scientific instruments. This arises owing to the control and data dependencies; these jobs are known as Grid workflows. The Grid workflows have many advantages, such as building dynamic applications which orchestrate and

utilise distributed resources, spanning the execution through multiple administrative domains so as to increase the processing capacity, and integrating different teams involved in managing different parts of the workflow [88].

- **Centralised and Decentralised Scheduling:** the difference between centralised and decentralised scheduling is in the control and the knowledge of the overall Grid resources. In the case of centralised scheduling, the scheduler has full control over resources, and the knowledge of the system is available by monitoring the resources state; thus, it is relatively easy to achieve efficient scheduling. The drawbacks of the centralised approach include limited scalability, which makes it inappropriate for very large-scale Grids, and the single point of failure. On the other hand, however, the decentralised scheduler has less control over the resources and much less knowledge, and therefore relies on local scheduler. The decentralised scheduler overcomes the drawbacks of the centralised scheduler, yet the decentralised scheduler could be less efficient than the centralised scheduler because the decentralised scheduler has less resources knowledge.

- **Immediate and Batch Mode Scheduling:** In immediate mode scheduling, the job is scheduled immediately after it arrives at the system. In batch mode scheduling, jobs are grouped together in batches and accordingly scheduled as a group. Importantly, batch mode scheduling takes better advantage of jobs and resources characteristics; therefore, batch mode scheduling achieves better resource utilisation by scheduling a batch of jobs rather than a single job. Immediate mode scheduling advantages can be seen in commercial Grid systems, when the Grid user requires an immediate answer to his/her SLA request.

- **Queuing and Planning Base Scheduling** In queuing-based scheduling, jobs are queued according to the scheduler policies, and the job begins executing when it arrives at the head of the queue and sufficient resources become available. Examples of queuing-based scheduling include LSF, PBS and Oracle Grid Engine [89] —previously known as Sun Grid Engine (SGE). On the other hand, planning-based scheduling requires the advanced

knowledge of the job execution time, and keeps track of the resources available, accordingly allocating a precise resource timeslot to every job. Examples of planning-based scheduler include Cluster Computing Software (CCS) [90]. Queuing-based schedulers follow the best-effort approach, and so there are no guarantees when a job will begin to execute. Planning-based schedulers do not suffer from this drawback as a job will be executed within a reserved slot, i.e. independent from other jobs. Queuing-based approaches are effective and easy to implement, although they produce less efficient scheduling than the planning-based approach, and they are also not suitable for immediate scheduling, which makes them inappropriate for commercial Grid systems. For a queuing-based scheduler to overcome these drawbacks, advance resource reservation is required. By using resource reservation, the queuing-based scheduler works as a planning-based one. One of the first attempts made in resource reservation was that of Maui [91], which is an external local resource manager, meaning it works in conjunction with a site's existing resource manager. It operates with all major local resource managers, such as PBS, LFS and LoadLeveller to extend their capabilities and subsequently enhance their scheduling effectiveness. Today, most of the queuing-based schedulers have advanced reservation capabilities, such as PBS and Oracle Grid Engine.

- **System-Centric and User-Centric Scheduling:** System-centric is a traditional scheduling approach which is commonly applied in single administrative domains by attempts to optimise system-wide measures of performance. System-centric Grid resource management systems, such as Legion [77] and Condor [78], adopt a conventional strategy where scheduling algorithms decide which jobs are to be executed at which resources based on functions driven by system-centric parameters. They aim to enhance the system throughput and utilisation, and to thereby complete execution at the earliest possible time. Notably, they do not take resource costs into consideration, which therefore means that the value of processing applications at any time is treated the same [92]. On the other hand, user-centric approaches concentrate on users' requirements by delivering maximum utility to the users of the system based on their QoS

requirements. For example, a guarantee of certain QoS based on the attributes that the user finds important, such as the deadline by which the job has to be completed. Enforcing the desired QoS requires a system of rewards and penalties; thus, it is common to find user-centric approaches driven by economic principles [92].

2.6.2.2 Predicting Execution Time

In the previous section, both planning-based scheduling and queuing-based scheduling with reservation assume the knowledge of all computational activities, such as required resources and execution time. This knowledge is assumed to be available from the resource user. This assumption is invalid, owing to the fact that most users do not have the time and experience to establish the required computational activities or otherwise make an accurate prediction about the required execution time. When users are asked to predict their application execution time, they tend to overestimate, which subsequently lowers the utilisation of the Grid resources. Systems which automate the prediction of the execution time will help the scheduling of jobs and the utilisation of Grid resources. Importantly, predicting execution time is an appealing subject which has been pursued by several studies [35, 93-101]. The approaches applied in such studies fall into two categories: learning-based approach or code-based approach.

The learning-based approach for predicting executing time assumes that applications with similar characteristics have similar runtimes; therefore, historical information from previous application runs are used in order to predict the execution time of future applications. Moreover, different learning algorithms can be applied on the historical information in order to predict the execution time. Kapadia *et al*. [94] evaluate the use of three local learning algorithms: nearest-neighbour, weighted-average and locally-weighted polynomial regression; they subsequently found that the simple nearest-neighbour algorithm outperforms the more complex algorithms. Furthermore, Dushay *et al*. [95] evaluate the use of three algorithms: running average, single last observation, and low-pass filter; they accordingly reached the same conclusion that simple prediction methods performed as well as more complex methods, and that prediction accuracy was closely related to data consistency. Djemame and Haji [35] evaluate the use of the three prediction algorithms presented in [95] so as to predict future run-time for the BROADEN

system. The low-pass filter algorithm outperforms the other two with the last observation algorithm being slightly less accurate. A recent study by Matsunaga and Fortes [98] evaluate six different learning algorithms: k-nearest neighbour, linear regression, decision table, Radial Basis Function network, Predicting Query Runtime and Support Vector Machine. They established that different algorithms perform better in different scenarios, and considering different configurations and algorithms is key to improving the quality of the prediction.

The code-based approach uses performance models reflecting application source code to provide performance estimates. An example of code-based prediction is available in the Performance Analysis and Characterisation Environment (PACE) [93]. PACE provides predictive information regarding execution time, system design and sizing, scalability and parallelisation strategies. Moreover, PACE analyses performance models, constructed from a performance language known as $CHIP^3S$, in order to achieve time-prediction. Other works include that by Brandolese *et al*. [96], which presents a methodology for the prediction of application execution time utilising a mathematical model derived from the source code.

Both learning and code-based approaches have advantages and disadvantages. The advantages of one provide the disadvantages of the other, and vice versa. In the case of the learning-based approach, predictions can only be made if historical information is available. Furthermore, historical information is crucial for the prediction process, and without consistence data, the execution time prediction will not be accurate. Another issue concerns the long time it takes to predict the execution time. On the other hand, code-based approaches do not depend on historical information, and the time that it takes to predict the execution time is minimal. However, there is the need to access the application source code, which is not always available, for example, because of copy writes. Moreover, application source code might need to be reengineered in order to be modelled.

2.6.2.3 Scheduling Algorithms

There exist many scheduling algorithms, and considering all of them will be a very long process; therefore, in this thesis, we consider only algorithms which address either BoT scheduling or scheduling with constraints. The reason for selecting BoT is that BoT jobs account for up to 96% of the CPU time consumed in Grid systems [87].

Maheswaran *et al*. [102] studied five immediate scheduling algorithms and three batch-scheduling algorithms for allocating BoT jobs to heterogeneous resources. The main objective of the algorithms is the maximisation of the throughput, with no requirements attached to BoT jobs, such as deadlines or costs.

Casanova *et al*. [103] considered three heuristic algorithms from [102, 104] for scheduling parameter sweep applications, also known as BoT applications (see Independent Scheduling in 2.6.2.1), and accordingly proposed an extension to one of the algorithms. The objective of the research is to take advantage of file-sharing so as to improve the performance of the algorithms. The aim of the algorithms is also to maximise the throughput and, as per the prior work, job constraints are not addressed.

Berman *et al*. [105] considered scheduling algorithms used in the Application Level Scheduling (AppLeS) project [106]. The main objective of the algorithms is to enhance the system throughput and utilisation; thus, job constraints are also not addressed.

Cirne *et al*. [107] propose the Workqueue with Replication (WQR) algorithm for BoT-scheduling. The algorithm is the same as a standard queuing-based scheduling algorithm; the only difference is that, when there are no BoT jobs in the queue, idle resources begin to execute a replica of an unfinished BoT job. The first replica to complete is the valid execution, whilst the other replicas are cancelled. The WQR was introduced in order to improve performance when information relating to the resources and BoT are not available. Job constraints are not addressed in the algorithm.

Lee and Zomaya [108] proposed the Multiple Queues with Duplication (MQD) algorithm for scheduling BoT jobs. The algorithm takes into account the recent workload pattern of resources in order to minimise the BoT makespan and to thereby maximise resource utilisation. Lee and Zomaya [109] also propose the Shared-Input-data-based Listing (SIL) algorithm, the main objective of which is to minimise data transfer, which will result in shortening the makespan of the BoT. Moreover, job constraints are not considered in both algorithms.

OurGrid is a middleware which facilitates the creation of P2P computational Grids [110]. Its aim is to speed-up the execution of BoT jobs. Notably, two different scheduling algorithms are proposed, namely Transparent Allocation Strategy [111],

which assigns jobs to idle resources, and Explicit Allocation Strategy [112], which assigns jobs to resources in order to reduce the turnaround time. Both algorithms do not take the job constraints into account, but merely focus on increasing resource utilisation. The algorithms have a simple approach for resource failure, which is re-executing the tasks affected by such failure.

The above algorithms are mainly system-centric. Their objectives are maximum throughput and utilisation. User-centric algorithms will address job constraints through the use of execution time-prediction. The benefits achieved in using user-centric rather than system-centric algorithms include the ability to address QoS, to optimise jobs requirements, to increase the performance, and to better utilise Grid resources so as to achieve QoS requirements.

Buyya *et al*. [113] consider scheduling parameter-sweep applications. Whilst the scheduling of these applications seems simple, complexities arise when users apply various constraints, such as deadline, total cost and quality of services. Four scheduling algorithms are proposed in an attempt to address only two constraints—deadline and budget. The scheduling algorithms are:

1. Time minimisation with limited budget (time-optimisation),
2. Time minimisation with unlimited budget,
3. Cost minimisation limited by a deadline (cost-optimisation), and
4. No minimisation, limited by a deadline and budget (no-optimisation).

It can be seen that Algorithm 2 is the same as Algorithm 1, but with very large budget and the algorithms became three. These algorithms were implemented in the Nimrod-G [114, 115] Grid resource broker and evaluated in [116]. The scheduling algorithms proposed—even with minimisation as a name—attempted to find a schedule which satisfies user constraints; however, it did not find a good minimisation—optimal or near-optimal—to better utilise the resources. Another limitation for such algorithms is that the minimisation only takes one constraint at a time and therefore cannot, for example, minimise time whilst simultaneously keeping costs at a minimum. Therefore, such scheduling algorithms are not sufficient enough, and better optimisation algorithms are required for this problem.

Buyya *et al*. [117] extended the aforementioned work by proposing a new scheduling algorithm for cost-time optimisation. This algorithm builds on the cost-

optimisation and time-optimisation scheduling algorithms, and takes into account the two constraints of time and budget. The cost-time optimisation algorithm is implemented within a simulator using GridSim toolkit [118]. Importantly, this algorithm has several limitations: firstly, the algorithm was only evaluated through a comparison with the cost-optimisation algorithm, which has its limitations; secondly, the algorithm does not consider the quality of the minimisation since it does not seek to establish an optimal or near-optimal solution; and finally, in arranging the resources, the algorithm takes into account only the cost of the resources—and only if two resources have the same cost is the resource performance considered.

Kumar *et al*. [119] mathematically modelled the cost-optimisation scheduling problem and state that it is not only strongly NP-hard, but is also non-approximable. A batch-scheduling algorithm is proposed for assigning BoT jobs to resources, minimising the cost and satisfying the user deadline constraint. A batch is made of BoT jobs, each with a deadline constraint and a penalty fee to be paid if it is not scheduled. The scheduling algorithm job is required to minimise the cost of allocating jobs to resources by maximising the number of jobs scheduled and minimising the penalty fee whilst also satisfying the deadline constraint. An optimal solution for this problem is feasible yet, for large problem instances, it will fail to provide a solution in a reasonable amount of time. Therefore, Kumar *et al.* [119] propose an efficient heuristic, known as Highest Rank Earliest Deadline (HRED), which is able to establish a near-optimal solution for a wide variety of problem instances very quickly. This algorithm has several limitations: firstly, the algorithm only considers optimising the costs, and it takes the deadline as the only constraint. Moreover, it does not address other problems, such as deadline optimisation. Secondly, it is a batch-scheduling algorithm which limits its use in the commercial Grid environments since commercial Grid users' require immediate response to their SLA, and an immediate scheduling algorithm is preferable. Finally, it is not a realistic scenario for a Grid resource provider to: (1) commit to all jobs; (2) run the batch-scheduling algorithm; and (3) pay the penalty fee for non-scheduled jobs. A more realistic scenario is: the Grid resource provider only commits to jobs that can be fulfilled and rejects the others without any penalty fee.

Macías *et al.* [120] propose an Economically Enhanced Resource Manager (EERM) for resource-provisioning based on economic models. The EERM is part of the Self-Organising ICT Resource Management (SORMA) project [121], which addresses the development of methods and tools for an efficient market-based allocation of resources through a self-organising resource management system. The overall aim of EERM is to isolate the SORMA economic layer from the Grid technical layer and thereby achieve maximum economic profit and resource utilisation by orchestrating and managing both economic and technical goals. EERM exists at each resource provider's site, and is designed to interact with a range of execution platforms (e.g. Condor, Oracle Grid Engine, or Globus GRAM). The scheduling algorithm in EERM merely focuses on enforcing the SLA requirements without any degree of optimisation. The current core SLA requirements include:

9. The number of CPUs, architecture and speed;
10. The type of Operating System, kernel version and shared libraries;
11. The Total free memory physical or virtual;
12. The total free local/network disk.

The EERM scheduling algorithm only enforces the SAL requirements, and does not have the ability to decide which jobs to accept or reject. A possible enhancement to the scheduling algorithm is to be able to optimise user constraints, such as cost or time, whilst enforcing the SLA requirements.

Menascé and Casalicchio [122] propose a simple QoS model for Grid-scheduling. Two constraints are time and budget, and three scheduling algorithms are proposed. The first scheduling algorithm minimises the job makespan without any consideration to cost; the second algorithm minimises the job makespan and satisfies the cost constraint; and the final algorithm minimises the cost and satisfies the time constraint. A limitation of this work is that it cannot be applied in real Grid environments owing to the assumption about the tasks: the work assumes a task can be divided and executed on more than one resource without any overhead, which is not the case in the real world. Another limitation is that the algorithms do not find an optimal solution—even though, under the previous assumption, finding an optimal solution in a reasonable amount of time is feasible.

Kurowski *et al.* [123] propose a new method for multiuser, multi-criteria job-scheduling in Grid environments. The work considers two constraints—time and cost—and users are able to express these as soft constraints as opposed to hard constraints. A batch-scheduling algorithm is used to establish a fair schedule of jobs submitted by multiple users; notably, schedule fairness is measured by user satisfaction. In a single-user scenario, the scheduler finds the solution that maximises the user satisfaction. In a multi-user scenario, the scheduler must find one solution which ensures a high satisfaction level for all users. Thus, the scheduling algorithm is more focused on modelling users' preferences and the evaluation of the extent to which a given schedule is satisfactory for each user, rather than optimising user constraints, such as minimising the cost or the makespan.

The user-centric scheduling algorithms above only consider the time and cost, and assume the resource price is a function of performance. A more expensive resource is always faster than a less expensive one; in the real world, this assumption is invalid. Ultimately, the reliability of the resource play a central part in the price: a more reliable resource is more expensive than a less reliable, even if the latter is faster. The reliability of a resource can be expressed as the resource risk of failure, and should be considered when scheduling. Unlike the above algorithms, the algorithm proposed in this thesis (Section 6.4) considers cost, time and risk of failure. Efficient heuristic is proposed in order to establish near-optimal solution in a reasonable amount of time.

2.6.3 Monitoring

Grid information services play a central role in Grid resources discovery (see 2.6.1). In order to fulfil this role, information services must collect information regarding the past and current status of Grid resources, which is known as monitoring. Data monitoring is also used in scheduling, performance analysis, performance tuning, performance prediction, optimisation of Grid systems, and many more; therefore, monitoring systems are of great importance since incorrect or out-dated resources information will hinder the Grid usage. Monitoring systems, according to Tierney *et al*. [124], should satisfy five requirements: low latency data transmission, high data rate, minimal measurement overhead, security, and scalable.

The MDS is not only used to discover resources in the Grid, but also to monitor these resources. It provides standard interfaces to query WSRF services for

resource property information. The MDS is not a monitoring system on its own, but rather provides interface connections to local monitoring systems and publishes summary data using standard interfaces. The Network Weather Services (NWS) [125] is a monitoring system that provides a short-term performance forecast based on historical information. NetLogger [126, 127] is another monitoring system that provides tools for generating event logs that capture resource and application information, as well as a Java interface to manage the large amount of logged data, and tools to visualise the data. NetLogger has four main monitoring components: the application instrumentation, the monitoring activation service, the monitoring event receiver, and the archive feeder. Ganglia [128] is a distributed monitoring system which can be used to monitor a single cluster or a federation of clusters through the use of point-to-point connection amongst different clusters. Ganglia was proposed with the objective to achieve low-node overheads, high concurrency and high scalability. Autopilot [129, 130] is an adaptive resource management system for dynamic application rather than a monitoring system, yet it uses sensors which capture application and system performance data. Autopilot sensors act as a monitoring system, recording data in a buffer before transmitting it. The data transmit can be on-demand, periodic, event-driven or conditional. For more information on Grid monitoring see [131].

2.7 Summary

This chapter has considered Grid computing and discussed the various types of Grid systems and applications running on them. The architecture facilitating the creation of such systems is presented, in addition to examples of projects which are implemented as Grid systems are given. Grid Resource Management activities—namely resource discovery, resource scheduling, and job execution and monitoring—are presented with a focus on resource scheduling. A survey of scheduling algorithms has been discussed alongside their benefits and limitations. Resource discovery, monitoring and job execution time prediction are also presented as requirements for job scheduling within Grid systems.

Chapter 3

Risk Assessment and Management

This chapter examines the definition of ‘risk’ in Section 3.1. Section 3.2 introduces risk analysis and approaches used. Section 3.3 defines risk management in general, and lists the three steps required, namely risk identification, risk assessment and risk response or treatment. Section 3.4 defines risk management in Grid computing, and provides a survey of models and techniques adopted in order to identify and assess risk. Finally, Section 3.5 summaries the chapter.

3.1 Definitions of Risk

There are several definitions of risk: for example, ‘the probability and magnitude of a loss, disaster, or other undesirable event’ [132] or ‘a measure of the potential loss occurring due to natural or human activities’ [9]. Regardless of the wording used to define the term, risk is nevertheless related to future events and their consequences. Notably, there is uncertainty associated with events and their consequences. The events uncertainty can be expressed by means of probability or likelihood, based on background knowledge [133]. It is important to distinguish between risk and opportunity:

- Risk is associated with events that, if occur, would have a *negative* consequences such as financial loss;
- Opportunity is associated with events that, if they occur, will have *positive* consequences.

Another important term linked to risk is ‘hazard’. Hazard typically refers to the source of the risk, i.e. risk is created by a hazard. For example, a toxic gas that is a hazard to human health does not represent a risk unless humans are exposed to it.

3.2 Risk Analysis

Risk analysis is the process of characterising and managing the potential events which may lead to negative consequences or losses. As with the definition of risk, different disciplines often categorise risk differently. Such categorisation can be carried out based on the events causing the risk or the consequences of such events. However, Modarres [9] categorises the risk into five broad categories: health, security, safety, financial and environmental.

Generally, there are three types of risk analysis: quantitative, qualitative and a combination of the two.

3.2.1 Quantitative Risk Analysis

The quantitative risk analysis attempts to estimate the risk in the form of the frequency of events and the magnitude of the losses or consequences. In this context, the 'uncertainty' associated with the estimation of the frequency of the occurrence of events and their consequences are characterised by using the probability concept.

Quantitative risk analysis is the preferred method when sufficient filed data, test data or other evidences exist so as to estimate the probability of events and magnitude of losses; however, quantitative risk analysis is complicated, time-consuming and expensive to conduct [9, 134, 135].

Quantitative risk analysis techniques includes: discriminate function analysis, Bayesian analysis, decision tree analysis, factor analysis, neural nets, risk matrix, risk register, and Mont Carlo analysis [8, 136, 137].

3.2.2 Qualitative Risk Analysis

Qualitative risk analysis is the most widely applied method, simply because it is simple and quick to perform. In this regard, the risk is estimated using a linguistic scale, such as low, medium and high. The frequency of events is measured by the likelihood of occurrence. In this type of analysis, a matrix is formed, which characterises the risk in the form of the likelihood of events versus the potential magnitude of losses in qualitative scale. This type of analysis does not rely on actual data and probability treatment of such data; accordingly, it is far simpler to use and

understand than the quantitative risk analysis, although it is extremely subjective [9, 134, 135].

Qualitative risk analysis techniques include brainstorming, assumption analysis, interviews, hazard and operability studies, and risk mapping. For a complete list, see [138].

3.2.3 Mixed Risk Analysis

Mixed risk analysis adopts a combination of qualitative and quantitative analyses. This mix can occur in two ways: either the frequency of an event is measured qualitatively, but the consequences are measured quantitatively or vice versa; or both the frequency of an event and the consequences are measured using quantitative methods, but the policy setting and decision-making are reliant on qualitative methods [9].

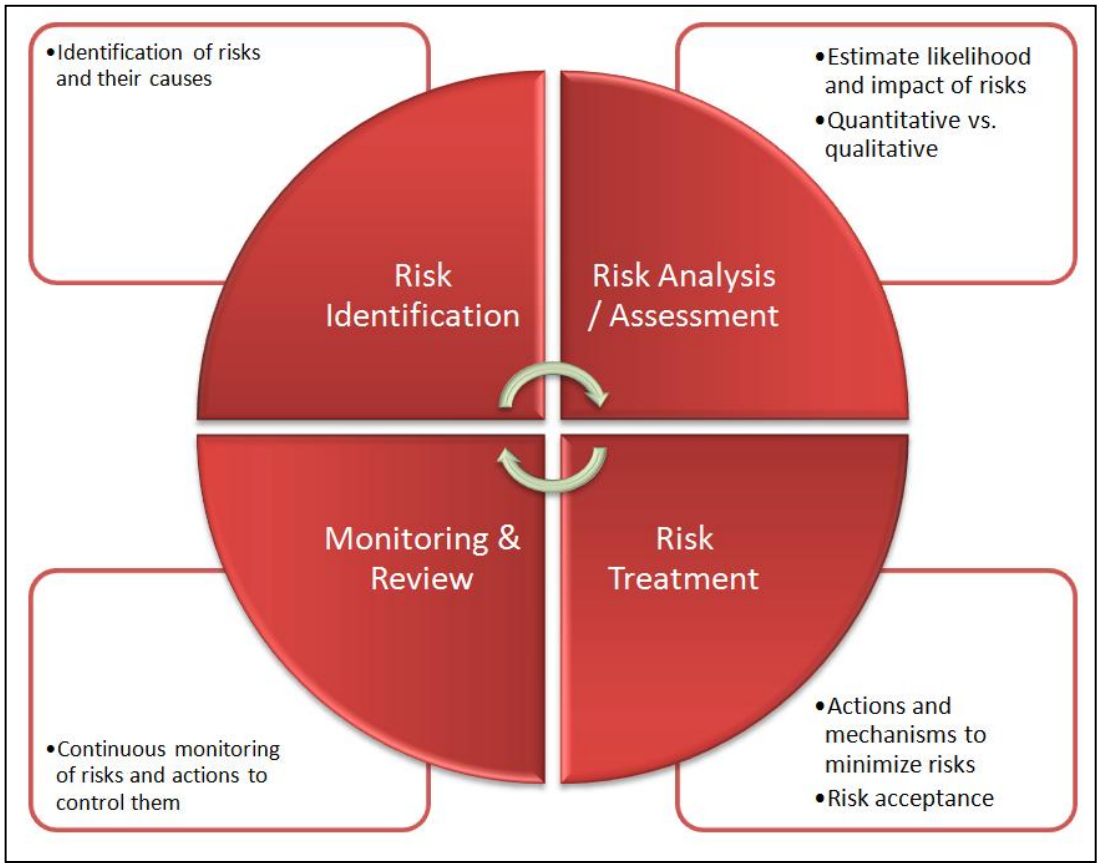


Figure 3: Risk Management Steps [139].

3.3 Risk Management

Risk management is the process that enables the identification, assessment, planning and control of risk [138]. Thus, the risk management process aim is

threefold: it must identify the source of uncertainty, assess the frequency of events occurrence and the consequences of those events, and respond to the risk in an appropriate and effective manner. The risk management is an iterative process and the identified risks are monitored throughout the lifecycle. Figure 3 shows the steps of the risk management process.

3.3.1 Risk Identification

The purpose of risk identification is to identify which risks are likely to occur, where risks may arise, what may be done in response to such risks, and what may go wrong with the responses. Both historical and current information are fundamental in the risk identification phase, and therefore should be analysed first. The identification of risk starts by analysing either the source of the problem or the problem itself. Importantly, sources could be internal, such as stakeholders or employees, or external, such as cultural differences or natural disasters. Problem analysis, on the other hand, identifies events or threats, such as losing money or damage repetitions which are not specific for one source but which ultimately arise from one or more sources. Essentially, there are a number of different methods for risk identification. The most commonly used risk identification methods are [138]:

- **Objective-Based Risk Identification:** Organisations and project teams have well defined objectives. They define risk as an event that may endanger achieving—partly or completely—one of their objectives.
- **Scenario-Based Risk Identification:** Different scenarios are created to represent alternative ways to achieve objective, and to accordingly analyse the interaction of forces in the environment. Any event that triggers an undesired scenario is considered a risk.
- **Taxonomy-Based Risk Identification:** This method presents a breakdown of possible risk sources according to certain criteria, and their degrees of importance. Based on the taxonomy and knowledge of best practices, a questionnaire is issued and the results are compiled [140].
- **Checklists:** In several industries, there are lists available with known risks. Each item in the list represents a threat, which can then be checked as to whether or not it applies in a particular situation. The lists take the form of either questions to be answered or a list of topics to be considered.

- **Risk Registers:** A risk register is a document or database that records each risk related to a project or assets. Risk registers from previous projects can be used to identify risk in the same way checklists are used.

3.3.2 Risk Assessment

Risk assessment is a set of techniques applied in order to investigate the probability of an event, and to thereby assess the effects/consequences of such [136]. Risk assessment is the most important phase in risk management: if the risk assessment method is not conducted appropriately, the risk management will then fail to achieve its objectives.

Selecting an assessment technique is not a straightforward task. According to the authors of [134, 135, 138], the selection of a technique viewed as most suitable for application on a process should be determined after considering the following:

- The availability of resources for analysis,
- the size and complexity of the process which will be analysed,
- the phase in which the risk assessment will be considered in the process lifecycle, and
- the availability of information.

The authors also emphasise the importance of the data considered in the risk assessment. The data considered should be accurate, adequate, relevant, coherent, unbiased and valid.

Regardless of the analytical techniques applied in the risk assessment, in order for the risk assessment process to be effective, various characteristics must be taken into account. According to Freeman *et al*. [141], the risk analysis must be:

- **Timely:** The process produces the best available data in an accepted time range.
- **Cost-Effective:** The cost of accomplishing a risk assessment is lower than the benefit gained from the results.
- **Complete:** The risk assessment must address all aspects of the process without taking anything for granted.

- **Consistent:** The methods used for evaluating risk and reporting threats must be consistent throughout the process.
- **Understandable:** The results must be communicated to the appropriate authority with clear terms.

3.3.3 Risk Response

Risk response is mainly concerned with what can be done in a situation after it has been assessed [142]. Once risks have been identified and assessed, some action must be considered in order to address each individual risk. The response usually falls into one of the following:

- **Risk Avoidance:** Risk avoidance involves the removal of the threat—either by eliminating the resource by redesign, more detailed design, or alternative development methods, or by otherwise avoiding any process which have exposure to risk. The later solution has a negative impact in terms of financial gain.
- **Risk Reduction:** Since risk combines the probability of events with the impact or the expected losses of those events, lowering the probability of an event, the consequence of the event, or both will ultimately result in risk reduction.
- **Risk Transfer:** Risk transfer is the process of transferring the risk to another party, who is able to bare the risk. Risk transfer does not eliminate or reduce the risk, but rather transfers the risk to another party to deal with the consequences. Insurance is a popular technique for risk transfer.
- **Risk Retention:** Risk may be retained intentionally or unintentionally. The latter occurs as a result of a failure, either in the risk identification phase or the risk assessment phase. Essentially, risk retention is a very good strategy when the risk is small, and the cost of responding to it is greater than the impact or the losses of it. All risks which are not avoided or transferred are retained by default.

3.4 Risk Management in Grid

The computer industry has expanded rapidly, and is one of the fastest growing industries at present. Computer systems are used in almost every aspect of life, such as industry, business, education, entertainment, health, defence, etc. As computer technologies continue to evolve, the risk of use similarly develops. Computer systems used in critical environments—such as nuclear power plants, air travel monitoring systems, medical devices, manufacturing processes, defence systems and stock exchange systems—need to be almost fault-free; a malfunction of such systems could be disastrous and might result in loss to devices, money or, even worse, life [143]. Therefore, risk management in such instances is of paramount importance. Other computer systems are less or non-critical, such as web servers or email servers, and the risk of faults of such systems is also lower than the risk of faults in critical systems. Nevertheless, a malfunction of non-critical systems might still result in losses of devices or money; therefore, risk management on such systems needs to balance between the cost of the risk management process and the expected loss as a result of faults. The risk management process cost should always be lower than the expected loss, otherwise it is more profitable not to implement such a process. Grid systems fall into the arena of non-critical systems (see Grid Applications 1.1).

Risk management can be carried out at various phases during the lifetime of a Grid system, i.e. from the development of a Grid infrastructure, through to the deployment and testing phase, right up to the operational phase. The rest of this section is devoted to review approaches adopted for risk management in computer systems in general, and Grid systems in particular.

3.4.1 Risk Identification

There are different sources of risks in Grid systems, depending on the systems phases: for example, in the development phase, there is a risk of software development failure; in the operational phase, there is a risk of hardware failure, information security breaches, etc. Each phase has various different risks associated with it, and events causing those risks need to be identified.

Software development projects suffer from a high failure rate [144, 145]. A number of risks identification checklists have been proposed [146-152]. Boehm

[146] identifies the top 10 risk items based on a survey of several experienced project managers. Schmidt *et al*. [147] identifies 53 risk items, and organises the list into 14 groups based on the source of the risk. The identification process was based on a three-phase Delphi survey the participants were made up of 41 experienced project managers from three different countries—USA, Finland and Hong Kong. Keil *et al*. [151] propose a framework for identifying software project risks whereby, instead of focusing on individual risk items, the framework provides four distinct types of software project risk, namely Customer Mandate, Scope & Requirements, Execution and Environment. All computer systems—not just Grid systems—suffer from risks related to software development.

The European Network and Information Security Agency (ENISA) [153] aims to be the European hub for the exchange of information, best practices and knowledge in the field of Information Security. In the context of ENISA's Emerging and Future Risk programme, 35 security risks of Cloud computing[1] [154] have been identified [155]. The identification process was based on the opinions of 22 experts from academia, industry and government. The risk items are organised into 4 groups: Policy and organisational risks, Technical risks, Legal risks, and risks not specific to the Cloud. The risks identified by ENISA are only related to information security.

The context-aware data-centric information sharing (Consequence) project [156] aims to deliver a data-centric information protection framework based on data-sharing agreements. A scenario where a group of organisations share data with each other but want to retain control over the usage of that data is used to identify the risks imposed on the security goals of confidentiality, integrity, and availability. Four critical security goals are identified: authentication, usage control decision, enforcement, and availability. Moreover, an attack tree is proposed in order to recognise sub-goals that must be achieved in order to accomplish any of the security goals. The consequence project only considers the risks of compromising the security of data shared in a distributed environment.

1 Cloud computing refers to both the software delivered as services over the Internet as well as the hardware and systems software that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS).

The SLA@SOI project [157] is to develop an SLA-aware service-oriented infrastructure, empowering the service economy in a flexible and dependable way. The project does not address risk specifically, although it does identify three factors relevant for reliability: software failures, hardware availability, and network failures [158]. Software failures and network failures are modelled in a probabilistic way, whilst hardware availability is modelled as the Mean Time To Failure (MTTF), divided by the sum of MTTF and Mean Time To Repair (MTTR). Other factors relevant for reliability are ignored in this project.

The main objective of the AssessGrid project [36] is to address obstacles of a wide adoption of Grid computing by bringing risk management and assessment to this field, thereby enabling the use of Grid technologies in business and society. In this scope, AssessGrid delivers generic, customisable, trustworthy, and interoperable open-source software for risk assessment, risk management, and decision-support in Grids. The AssessGrid project applies a scenario-based risk identification approach [159], and identifies two risk items: the risk to the resource provider, and the risk to the broker. The risk to the resource provider is the violation of users' SLAs, which is influenced significantly by resources failure. A source analysis is used to identify the resource failure, which can be internal, such as hardware failure, problems in software components, version problems in used software systems, power supply problems, etc., or external, such as no delivery on external contracts, natural disasters, etc. The risk to the broker is the unreliable methods used by the resource provider to assess the risk of failure. The broker plays a mediator role between Grid providers and users: its primary task includes the assignment of the user jobs to certain resource providers in order to minimise the overall possibility of failure in carrying out those jobs. Importantly, the broker aims to minimise the aggregate risk of failure of all tasks under its management.

3.4.2 Risk Assessment

A fundamental concept in risk assessment is the concept of Risk Exposure (RE), sometimes referred to as risk impact [160]. RE is defined as:

$$RE = Prob\ (UO) * Loss\ (UO)$$

where Prob (UO) is the probability of an unsatisfactory outcome and Loss (UO) is the loss to the parties affected if the unsatisfactory outcome occurs. RE is then used to produce a ranked ordering of the risk items identified.

In consideration of software development projects, the probability and the loss of an unsatisfactory outcome are assessed via application of the qualitative risk analysis technique. Boehm [146] proposes the use of a scale 0–10 in order to assess the probabilities and losses of unsatisfactory outcomes; such assessments are often the result of surveying several domain experts and are frequently subjective. Furthermore, there is some uncertainty in terms of estimating the probability or loss associated with an unsatisfactory outcome, which is, itself, a major source of risk. Keil *et al*. [151] adopts a three-phase Delphi survey in order to immediately identify the most important risk items, rather than simply identifying probability or loss associated with an unsatisfactory outcome. The survey identified 11 risk items as the most important.

The aim of this survey is to serve as a checklist of the most important risks for project managers to focus on. Wallace and Keil [150] map the 53 risk items identified in [147] into the four risk categories proposed in namely Customer Mandate, Scope & Requirements, Execution and Environment. A survey of 507 project managers, representing multiple industries, indicated the extent to which each risk item was present during their most recently completed projects. A scale from 1–7 is utilised so as to represent the presence of a risk item; higher numbers represent a higher presence and lower numbers a lower presence. The result identifies the risk associated with the Scope & Requirements and Execution categories to be the most critical, and that the Environment category is not of great importance.

The qualitative assessment of the 35 security risk items identified by ENISA in [155] is based on three scenarios: Small and Medium Enterprises (SME) migration to cloud computing services, the impact of cloud computing on service resilience and cloud computing in e-Government. The risk assessment is based on the ISO/IEC 27005:2008 information security risk management [161]; the risk is estimated on the basis of the likelihood of an incident scenario and the negative impact of that scenario; and the likelihood and the negative impact of a scenario are estimated using the following scale:

- 0, or Very Low,
- 1, or Low.
- 2, or Medium,
- 3, or High,
- 4, or Very High

The likelihood and the negative impact are determined by several domain experts. The risk is measured as the sum of the likelihood and the impact.

$$Risk = likelihood + impact$$

The risk is mapped to a simple risk rating Low Risk 0-2, Medium Risk 3-5 and High Risk 6-8. This qualitative risk assessment is based on surveying several domain experts and might be subjective. Furthermore, there is some degree of uncertainty in terms of estimating the likelihood or the negative impact, which is, itself, a major source of risk.

The objective of the Consequence project [156] is to provide an information protection framework and to thereby identify the security risk in sharing data in a distributed environment. The risk items are used as a checklist of items to be addressed in the Consequence architecture, without any assessment of the probability and the negative impact of a risk item.

The SLA@SOI project [157] does not explicitly address risk assessment, although it does propose the utilisation of a prediction service for estimating the probability of software failure, hardware availability and network failure in an attempt to evaluate the QoS. The work on the prediction service is in its early stage, and results are expected later in 2011 [158]. Notably, even in this early stage, a number of limitations can be identified. The hardware availability is defined as:

$$\text{Hardware Availability} = MTTF / (MTTF + MTTR)$$

This availability is for the entire lifecycle of the hardware, and it is not the probability that a hardware resource is available just at the point in time when it is required by service execution as assumed in the prediction service [158]. Another shortcoming is that the hardware might be unavailable owing to software failure or network failure; this means a single failure is considered twice in the analysis. Finally, the prediction service is not able to aggregate the probability of software

and network failure to predict the probability of system failure as other components affecting the system failure are not addressed, i.e. hardware failure, electricity outage, air conditioning failure, etc.

The AssessGrid project [36] determines the probability of an SLA failure as:

> Probability of (*n* nodes will fail for the scheduled duration of a task) × 1 - (the probability of (*m* reserve nodes are available for the scheduled duration of a task)).

The probability of node failure is calculated by assuming that the node failures represent a Poisson process[1], which is non-homogenous in time and has a rate function $\lambda(t),\ t > 0$ [162]. Many studies assume that the failure rate follows a Poisson process [163-165], although there is strong evidence to support that this is not the case [163, 166, 167]. Another limitation of the Poisson process assumption is that the repair time is either neglected completely or otherwise follows a Poisson process. The determination of the distribution for $\lambda(t)$ in AssessGrid is based on the Possibility theory, as initiated by Zadeh in [168]. It assumes that Grid failure data are rarely available, and recording such failures is infrequent; therefore, probability theory models cannot be used. With this in mind, possibility theory is based on new concepts: possibility measure, necessity measure, possibilistic distributions, etc. The parameter estimates are based on Gamma distributions and builds on a family of Bayesian models. The subjective selection of the prior distribution in Bayes Theorem might violate the objectivity of failure analysis.

The AssessGrid broker provides information that supports the end-user in terms of evaluating the reliability of providers' risk assessments. For each accepted SLA, the broker stores the details in a database, including the final status (Success or

---

1 A Poisson process is a continuous-time counting process ($N(t),\ t \geq 0$) that possesses the following properties:

- $N(0) = 0$
- Independent increments (the numbers of occurrences counted in disjoint intervals are independent from each other)
- Stationary increments (the probability distribution of the number of occurrences counted in any time interval only depends on the length of the interval)
- No counted occurrences are simultaneous.

Fail) and the offered Probability of failure PoF. The reliability of the providers' risk assessment is computed by comparing the number of observed failures with the number of failures predicted by the provider's offered PoFs normalised by the predicted failures standard deviation [169].

Resources failure plays a fundamental role in assessing risk in the Grid operational phase. Estimating the frequencies of failures must be through quantitative methods, as:

- resources failure data are available, and
- experts have no means to specify the likelihood of such failures.

Therefore, the next subsection is dedicated to grid resource failures.

3.4.2.1 Grid Resources Failures

A large number of studies that look at resource failure are found in the literature, including [170-176]. Schroeder and Gibson [170] analyse failure data collected over 9 years at Los Alamos National Laboratory (LANL), and includes 23,000 failures recorded on more than 20 different systems—mostly large clusters of Symmetric-Multi-Processing (SMP) and Non-Uniform-Memory-Access (NUMA) nodes. The source of a failure falls in one of the following: human errors and environments, such as power outages, hardware failure, software failure, network failure and unknown failures. They find that the time between failure at individual nodes—as well as at an entire system—is fit well by a gamma or Weibull distribution with decreasing hazard rate (Weibull shape parameter of 0.7–0.8). The observation that the time between failures is best fitted by a Weibull distribution with decreasing hazard rate is evidence in the studies [171-175]. Iosup *et al*. [176] consider the availability of CPUs in a Grid environment and analyse availability traces recorded from all the clusters. The finding is that the best fit distribution is Weibull with a shape parameter > 1. The reason for that is that many of today's Grids comprise computing resources grouped in clusters, the owners of which may share them only for limited periods of time. Often, many of a Grid's resources are removed by their owner from the system—either individually or as complete clusters—in order to serve other tasks and projects; thus, the unavailability of CPUs is not owing to a system failure but rather their unavailability by their owner. Most of the previous studies considered only short-term availability data [173, 174]. Other

studies used statistical modelling to predict failure at Grid level not resources level [175]. Importantly, these studies only consider distribution fitting to the failure data. This approach does not take into account the effect of system repairs, and also only assigns the probability of first failure at time *t*.

Another approach for predicting the probability of resource failure without assuming that the resource failures represent a Poisson process is by computing the resource availability. The availability function A(t) of a resource is the probability that the resource is operational at the instant of time *t*. Therefore, the probability of resource failure at time *t* is 1 – A(t). On the other hand, the reliability function R(t) of a resource is the conditional probability that the resource has survived the interval [0,t], given that the resource was operational at time *t*= 0. Availability differs from reliability in that any number of resource failures can have occurred before time *t*. Reliability is used to describe resources in which repairs cannot take place, as in satellite systems, resources that provide critical functionality and cannot be down even for repairs as in aircraft systems or resources in which the repair is extremely expensive. Generally, it is more difficult to build a highly reliable resource than a highly available one [177].

Nadeem, Prodan & Fahringer [175] propose a model to predict the availability of three different Grid resources: dedicated resources which are always available to Grid users, temporal resources which are available to Grid users as long as they are switched on, and on-demand resources which are only available to Grid users by demand. The models proposed are building on Bayes Theorem, and predict the availability as a function of day-of-the-week and hour-of-the-day. This approach has a number of limitations: for example, it does not differentiate between the unavailability as a result of node failure and the unavailability as a result of scheduled maintenance or repair; secondly, the models only consider the hour-of-the-day, and so a 1-minute unavailability and 1-hour unavailability are treated the same—even worse if the unavailability falls at the end of an hour and into the beginning of the next, and the unavailability subsequently becomes 2 hours.

Another approach to model system availability and reliability in computing is through the use of Markov models. Hacker, Romero & Carothers [178] investigated the use of Semi- Markov models to model node reliability in relation to large supercomputing systems. Platis *et al*. [179] adopt a two-phase cyclic non-

homogeneous Markov chain with the objective to evaluate the performance of a replicated database. Koutras, Platis & Gravvanis [180] explored the use of homogeneous continuous time Markov chain with the amount of free memory to model the resource degradation of a computer system. Furthermore, Koutras, Platis & Gravvanis [181] studied the use of a cyclic non-homogeneous continuous time Markov chain in terms of driving an optimal software rejuvenation model.

The probability of resource failure plays a central role in the risk assessment process. The above models to compute this probability have some limitations: the unrealistic assumption that the resource failures represent a Poisson process, the subjective prior distribution selection in the Bayesian model or ignoring resource unavailability due to scheduled maintenance. Therefore, this thesis proves that the resource failures does not represent a Poisson process (Section 4.4), fit distributions to observed resource failures data (Section 5.2), and model the resource using the Markov model technique to represent all the resource states and address the scheduled maintenance (Section 5.3).

3.4.3 Grid Risk Response

Risk response is outside the scope of this thesis, since this thesis focuses on the most important step in the risk analysis which is risk assessment (chapter 4 & 5). Yet in this section an overview of the risk response is presented to increase the reader's knowledge.

The risk to software development projects—as well as the risk to information security—is usually treated at the design phase. The aim is to lower both the likelihood and the impact of an undesired event. The Software Engineering for Service-Oriented Overlay Computers (SENSORIA) project [182] provides tools to enable developers to model their Grid applications at a very high level of abstraction with the use of service-oriented extensions of the standard UML, or domain-specific service-oriented modelling languages to translate into hidden formal representations by automated model transformations. Furthermore, such tools are able to perform early performance analysis, check the functional correctness of services, and accordingly predict the bottlenecks in collaborating services.

The responses to the risk of resources failure are to lower the probability of the failure or to lower the impact. The probability of failure can be lowered by investing

in new infrastructures, advanced monitoring services, and experienced system administrators, etc. Importantly, the impact of a failure can be lowered through the use of fault-tolerance mechanisms, such as reserve idle resources and checkpointing. Checkpointing is the process of periodically saving sufficient information about application or resource state to avoid having to restart the application from the beginning [183]. The advantage of combining the checkpointing with PoF is that checkpointing will be carried out frequently in relation to those resources with high PoF, and less frequently concerning those resources with low PoF. This will lower the overheads on reliable resources. The benefit of checkpointing exactly before the point of failure are presented in [184].

3.5 Summary

This chapter has considered risk management and discussed the types of methods for risk identification, assessment and response. Examples of risk items identified are provided. A survey of the risk assessment methods for software development projects, information security and resource failure have been discussed alongside their benefits and limitations. Finally, the response to risk is presented.

Chapter 4

Analysis of Failures in Grid Environments

In this chapter, the motivation scenario for introducing risk assessment method in order to improve the commercial uptake of Grid computing is showcased. The events causing risk have been identified, and the measurement of risk is introduced. The analysis of Grid resource failures is presented in detail, following the data collection. The statistical proprieties of the data—including the root cause of failures, the mean time to repair and time between failures—are also analysed. Finally, the resource failures are tested against well-known probabilistic failure models in order to verify whether they can be used to model the Grid resources.

4.1 Motivation Scenario

Over the recent years, the use of Grid computing has become the alternative to the traditional tightly coupled computer systems. Grids provide cost-effective and easily scalable resources, although the commercial uptake of Grid computing has remained slow. Current Grid middleware (e.g. Globus Toolkit) still follows the best-effort approach; there is a risk that users do not get any guarantee that their SLA will be fulfilled. Furthermore, Grid resource providers are not attracted either: for a resource provider, agreeing on an SLA without enough information about the state of resources and the availability of devices introduces a chance of violating the SLA, which can then result in a penalty fee. Furthermore, there is a risk attached to system failure, service unavailability, insufficient resources, etc., which might lead to SLA violation. Importantly, without a method for assessing the risk of accepting an SLA, providers are only able to make uncertain decisions regarding suitable SLA offers. Furthermore, users would like to evaluate the risk of a provider violating an SLA so that they are able to make decisions concerning which Grid resource provider to select and the acceptable cost/penalty fee associated with the SLA.

Figure 4 illustrates the motivation scenario, and demonstrates how risk assessment fits in the use of Grid systems. The user submits an SLA request to the resource provider. The SLA includes the user's requirements, such as deadline or cost. When the resource provider receives the SLA request, it contacts the resource reservation component to reserve the end user required resources within the deadline requested. If resources are not available, the SLA is rejected; otherwise, for each resource, the time *t* in which the reservation starts and the duration *d* are sent to the risk assessor.

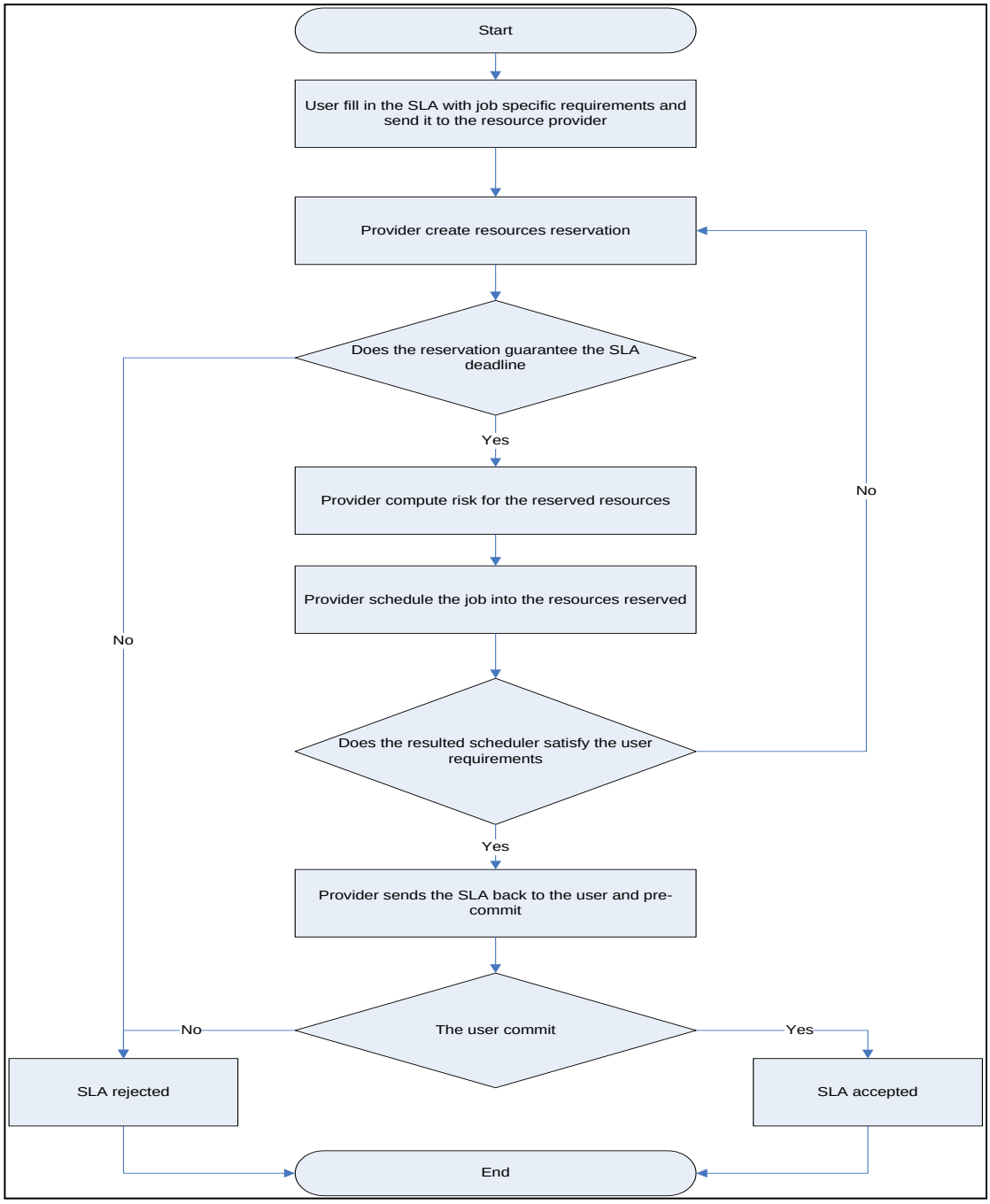


Figure 4: Flow Chart of the Motivation Scenario.

The provider computes the risk for each resource and subsequently allocates the resources to the user job. If the resulted allocation fails to satisfy the user requirements, resource reservation is revisited; if it does satisfy the user requirements, the provider then sends back the SLA, updated with cost and penalty fee and pre-commit. The user either commits to the SLA or rejects it. Figure 5 provides an overview of components in the resource provider infrastructure. The user sends an SLA request to the provider with the job requirements (1). The provider's Resource Manager requests the Reservation & Allocation component to reserve the required resources (2). The Reservation & Allocation component reserves the physical resources (3) and forward for each reserved resource the time and duration of the reservation to the Risk Assessor (4). The Risk Assessor computes for each resource the risk of failure based on the resource historical information stored in a database (5). The Monitoring component is responsible for updating the information in the database. The Risk Assessor returns the risk of failure information to the Resource Manager (6). Finally, the Resource Manager sends the SLA response back to the user (7), either accepting or rejecting the SLA.

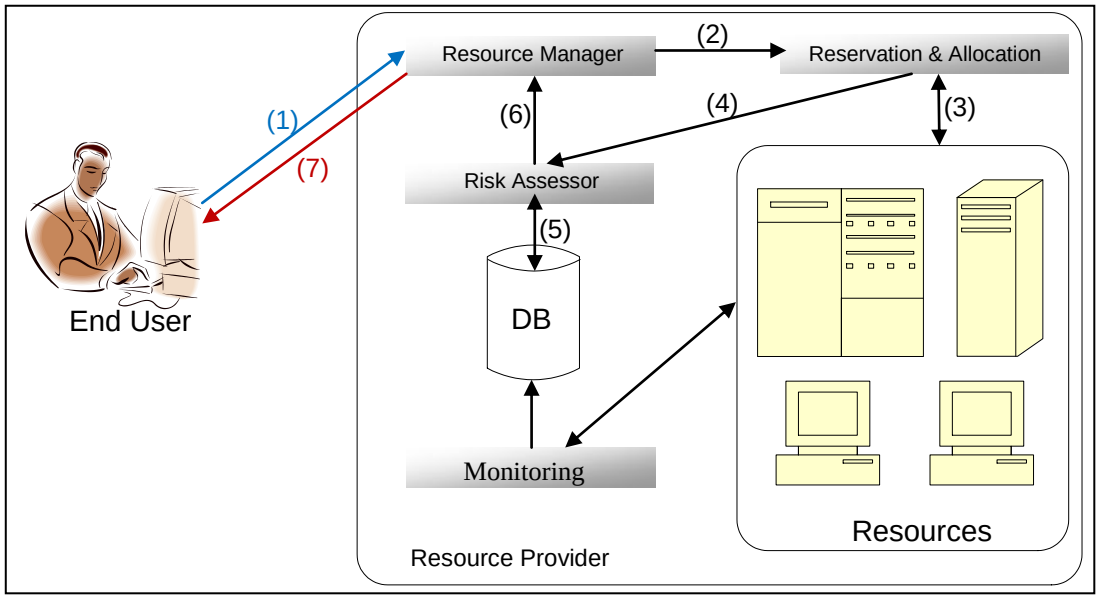


Figure 5: Overview of Components in Resource Provider.

The scenario highlights two components in the field of Grid computing which is currently suffering from limitations: a risk assessment method (see 3.4.2) and a risk aware resource allocation (see 2.6.2.3). The rest of this chapter is dedicated to the risk assessment methods, while the risk aware resource allocation is discussed in chapter 6.

4.2 Risk Identification

The definition and representation of risk can vary between different fields, as highlighted in Chapter 3, and so it is therefore very important to define risk in the context of Grid computing. In Grid computing, the assets are the Grid resources, the risk failures of which is of great concern. This thesis, investigate the risk of Grid resources failures (ROF). In order to correctly specify the ROF the probability of the resource failures and the impact of the failures need to be identified.

In order to compute the probability of resource failures, the events which cause a resource to fail first need to be specified. Grid resources can fail as a result of a failure of one or more of the resource components, such as CPU or memory; this is known as hardware failure. Another event which can result in a resource failure is the failure of the operating system or programs installed on the resource; this type is known as software failures. The third event is the failure of communication with the resource; this is referred to as network failures. Finally, the last event to cause a resource failure is the disturbance to the building hosting the resource, such as a power cut or an air conditioning failure; this type is known as environment failures. Sometimes, it is difficult to pinpoint the exact cause of the failure, i.e. whether it is hardware, software or network failure; this is therefore referred to as unknown failures.

4.2.1 Probability of Resource Failure

A set E_H is used to denote the events which cause hardware failures, and $P(E_H)$ is the probability of such hardware failures, where E_S denotes the events that cause software failures and $P(E_S)$ is the probability of software failures. Notably, E_N denote events that cause network failures and $P(E_N)$ is the probability of network failures, E_E denotes events that cause environment failures, and $P(E_E)$ is the probability of environment failures. Finally, E_U denotes events which cause unknown failures whilst $P(E_U)$ is the probability of unknown failures. These sets of events represent the complete events, denoted as E, that cause a resource failure. Thus:

$$E = (E_H \cup E_S \cup E_N \cup E_E \cup E_U)$$

The probability of resource failure is:

$$P(E) = P(E_H \cup E_S \cup E_N \cup E_E \cup E_U)$$

Recall that an important consequence from the probability axioms is [185]:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

The sets E_H, E_S, E_N, E_E and E_U are disjointed (or mutually exclusive), i.e. if the resource fails at a given time *t*, then only one event from the set could have caused this failure. In an extreme case, two events from different sets might take place at one time, yet the person responsible for repairs will only identify a single event. Therefore:

$$\forall I, J \in \{H, S, N, E, U\} \& I \neq J$$

$$E_I \cap E_J = \emptyset$$

From the probability axioms:

$$P(\emptyset) = 0$$

Therefore the probability of resource failure is defined as:

$$P(E) = P(E_H) + P(E_S) + P(E_N) + P(E_E) + P(E_U)$$

4.2.2 Impact of Resource Failure

The impact of resource failures is not as straightforward as the probability of failures as both resource providers and resource users have competing needs. For resource providers, resource failures have a financial impact in the form of penalty fee and, if the resource provider has a reputation system[1] [186], a reputation impact in the form of negative review or feedback from the unsatisfied user. Even in the absence of a reputation system, unsatisfied users might put forward their negative experiences to friends or co-workers, write about them in blogs or internet forums, or review the provider services on review sites, such as www.epinions.com.

The impact of resource failures on users is very hard to compute. Different users have various different requirements. For example, after a resource failure, User A might use another available resource to redo the work without any impact, whilst

[1] A reputation system collects, distributes, and aggregates ratings and opinions about participants' past behaviour and dynamically compute the reputation scores.

User B has a deadline to meet, and the resource failure would mean User B misses it. Even with the payment of the penalty fee, the financial loss owing to the missed deadline might be greater than the penalty fee.

4.2.3 Risk Measures

It has been pointed out in Chapter 3 that the word 'risk' is used to combine the probability of events with the impact of those events. Whilst computing the probability of the resource failures is feasible, computing the impact of failures is difficult, problematic and complicated—even if only the financial impact of failures is considered. The reason for this is that resource providers and resource users have competing needs; thus, a resource provider would need to set a low penalty fee in case of a resource failure, and the user would require that the penalty fee be high. Another problem is that resource providers and users have different views of risk. To illustrate this point, an example is provided below.

Assume that a user requests a resource to use from a resource provider for a period of time, starting from 12:00 o'clock. The provider computes the probability of failure for the resource for the period [12:00, (12:00 + t)] as X. The impact of the failure is linked with the penalty fee; thus, the provider can lower the impact by lowering the penalty fee. Consider that the risk is lessened by either reducing the probability of the event, the impact of the event, or both (see 3.3.3). Therefore, the provider can reduce the risk by lowering the impact—despite the probability X remaining unchanged. For the user, lowering the penalty fee increases the impact, and so the risk to the user is increased rather than decreased when the penalty fee is lowered. The actions that reduce the risk to the resource provider increase the risk to the resource user and vice versa. On the other hand, however, decreasing the probability X will reduce the risk for both the provider and the user.

The above example shows that the impact of failure has an opposite effect on the provider and the user, whilst the effect of the probability of failure is the same for both parties; therefore, it is more appropriate to measure risk to both parties only in terms of the probability of failure. This type of measurement is consistent, since resource providers and users have the same view on the probability of failure. As a result, in this thesis the ROF is defined as:

$$ROF = P(E)$$

The ROF formula above is limited to only the probability of resource failure. Even though there are qualitative approaches to compute the impact on both the user and the provider, the impact is neglected. This is because the qualitative approaches are outside the scope of this thesis.

4.3 Grid Resource Failures

Analysing the Grid resources failures and understanding the performance of those resources with time is a key requirement for their modelling. Therefore, in this section, the need for resource failure data and the collection process is presented along with the methodology used to analyse the data. Three metrics are studied: the root cause of failure, the repair time, and the time between failures.

4.3.1 Failures Data Collection

Gathering information relating to the past and current status of Grid resources—known as monitoring—is an essential activity. Monitoring data is used in the case of scheduling, performance analysis, performance tuning, performance prediction, the optimisation of Grid systems, and many more (see 2.6.3 for information about monitoring and monitoring tools). Monitoring resource failures is crucial in the design of reliable systems, e.g. the knowledge of failure characteristics can be used in resource management to improve cluster availability [172]. Creating realistic benchmarks and test-beds for reliability testing requires the knowledge of failure characteristics [170]. Furthermore, calculating the probability of failure of a resource depends on the past failures of a resource; therefore, access to resource failures data is very important.

Importantly, the resource failures data should be complete in the sense that all failures are reported, and also consistent in the sense that the reporting procedure is the same and span for a long time. These factors should be ensured for two reasons:

- A large number of failures observed will smooth out random variations and will result in a reasonably good probability estimation; and
- Long time observation reflects the true behaviour of resources.

Resource failures data that satisfy the above requirements are not easily available, and data collected in academic institutes might be incomplete or

inconsistent. Furthermore, commercial institutes are usually reluctant to share their data..

The Grid Operations Centre Data Base[1] (GOCDB) [187] is the official repository for storing and presenting European Grid Infrastructure (EGI) [26] topology and resources information. GOCDB stores information for all sites within the Enabling Grids for E-science (EGEE) [23], the National Grid Service (NGS) [25] and Worldwide LHC Computing Grid (WLCG) [188]. The stored information can be classified into six main groups: Users, Sites, Nodes, Services, Groups and Downtimes. GOCDB is publicly available and accessed following registration.

A user in GOCDB either has read-only access to all the public features or has a role to add, delete or edit information. A role is assigned to a user following a registration, and a single user may have one or more roles assigned. Roles fall into three categories: site level roles, regional level roles, and project level roles. For a complete list of roles and permissions associated to them, see [189].

A site is a physical location—such as the European Organisation for Nuclear Research (CERN) [190] or the Grille de Recherche d'Ile de France (GRIF) [191]—containing Grid resources. Thus, a Grid provider is represented as a site in GOCDB. The site's information stored in GOCDB are identification (ID), short name, official name, domain name, home web URL, contact email and telephone number, security contact email and telephone number, hours of operation, time zone, site's Grid Information Index Server (GIIS) URL, whether or not the site a primary site, description, the latitude and longitude, country in which the site is located, firewall IP address and the ID of the user who created the site and the creation date.

A node is a computer providing Grid services. Therefore, a Grid resource is represented as a node in GOCDB. In this thesis, the words 'Grid node' and 'Grid resource' are interchangeable. The nodes information stored in GOCDB are ID, hostname, IP address, host certificate Distinguished Name (DN), description, whether or not the node is a core node and a list of services running on the node.

[1] The selection of the failures data source was based on emails exchange with NGS support.

A service represents Grid software that provides a Grid service to the infrastructure, such as storage or processing capacity. Each node provides one or more services, and the service type must fall into a predefined set of services, e.g. Storage Resource Manager (SRM) or gLite Workload Management Service (WMS). For a complete services list see [192].

A group is a collection of sites grouped together. GOCDB stores the group name, a description of the group, type of the group and a contact email. Unlike other information, groups cannot be added to GOCDB through the input system web interface, but requires the involvement of a GOCDB administrator. For group registration procedure, see [193].

A downtime is a period of time for which a grid node is declared to be inoperable. A downtime record contains unique downtime ID, downtime classification (scheduled or unscheduled), the severity of the downtime, the user who recorded the downtime, the date at which the downtime was added to GOCDB, the start and end of the downtime period, the description of the downtime, and the entity affected by the downtime. (For a downtime sample see Appendix A).

Scheduled downtimes are planned and agreed in advance, whilst unscheduled downtimes are unplanned and are usually triggered by an unexpected failure. EGEE defines specific rules [194] concerning what should be classified as scheduled downtime and what should be classified as unscheduled downtime. The rules are based on the length of the intervention, the impact severity, and how long in advance the downtime is declared. These rules were later relaxed to one rule: a scheduled downtime needs to be declared 24 hours in advance, otherwise it is automatically declared as unscheduled downtime. Unscheduled downtimes should be declared as soon as they are detected; however, they can be reported up to 48 hours following the downtime [195].

The severity of the downtime is either ‘at risk’ (whereupon the resource will probably be working as normal, but may experience problems) or ‘outage’ (whereupon the resource will be completely unavailable).

The user whom has permission to make downtime updates can add, edit, or delete downtime information; this is done manually, and there are no rules or protocols to make such updates. Accordingly, it might be possible that the resource encounters a failure, and that there is no record on the GOCDB for such failure.

The description of the downtime is left to the Grid administrator; it is a short description of the cause of the downtime. Importantly, there are no rules or protocols to follow when writing the description; thus, descriptions are mostly incomplete and are very ambiguous, with some possibly having only one very brief word description (e.g. Test).

The downtime data collected in GOCDB is different compared with the data in error-logs. Error-logs are generated automatically, and treat every unexpected event the same—whether or not it resulted in a system failure. Also, error-logs might contain multiple entries for the same event; on the other hand, however, downtime data in GOCDB are created manually by system administrators. Human created failure data have two potential problems: underreporting of failure events and misdiagnosing the cause of the downtime. Although it is possible for a failure not to be reported at all, in this thesis, we are assuming that this is not the case; misdiagnosing the cause of the downtime is feasible. GOCDB dose not have classification of the root cause (e.g. Hardware, Software, etc) it has only a description of what might cause the downtime. The diagnosis and description depend hugely on the administrators' skills.

In this thesis, we take into account the downtime data for seven Grid resources (or nodes) from two different Grid sites. Four resources are from Site 1, and three resources are from Site 2. We name Site 1 resources A, B, C, and D, and Site 2 resources A, B and C. The reasons for selecting these resources are:

- Different resources and sites are used to generalise the findings; otherwise, the finding will be limited to a specific resource or site;
- The resources considered join GOCDB in its early stage and frequently record downtime data;
- Since the description of the downtime is left to the Grid administrator, some descriptions are ambiguous or incomplete. Therefore, the selected resources have comprehensive downtimes description.
- Resources frequently join and leave the Grid; therefore, the selected resources never leaved the Grid;
- The selected resources offer different Grid services; and

- The downtime data for all resources span for three years from the start of 2008 till the end of 2010.

The downtime data have scheduled and unscheduled downtime, but we only consider unscheduled failures. The reason for this is that the uses of advance reservation takes into account scheduled downtimes.

4.3.2 Methodology for Failure Analysis

A resource is considered to be a failed resource when it is not performing as normal. Therefore, a resource declared in GOCDB as ‘at risk’ or ‘outage’ is considered to be a failed resource.

In the next sections, we will analyse resource failure data with respect to three important properties of system failures: root cause (4.3.3), time to repair (4.3.4.), and time between failures (4.3.5). Moreover, the sequence of failure events are studied using stochastic process [196] and the distribution of its time between failures is also considered. Notably, we characterise repair times for each resource using the mean, median and standard deviation. We also consider the empirical cumulative distribution function (cdf) of repair time for each resource, as well as how well it fits four probability distributions commonly used in reliability theory: Exponential, Weibull, Gamma and Lognormal distributions. These distributions fit the data well, and so there are no reasons for using other distributions or more degree of freedom e.g. a phase-type distribution. Notably, we utilise the Maximum Likelihood Estimation (MLE) to parameterise the distributions and thereby evaluate the goodness of fit by visual inspection, and the negative log-likelihood test. The MLE—unlike moment estimation—is consistent, unbiased and efficient [10]. The cdf for the time between failures for each resource is analysed also using MLE and the negative log-likelihood test.

4.3.3 Root Cause Breakdown

The first question to ask when studying failures in computer systems is “what caused them?” In GOCDB data, there is a description of the cause of failure; however, there is no classification for such causes. We are therefore required to map the description of the failure into five different categories: Environment, Network, Software, Hardware and Unknown. Figure 6 shows the percentage of failure in each category for Site 1. The right-most bar highlights the breakdown of all the failure

recorded in Site 1, whilst the first four bars are for resources A, B, C and D respectively. Figure 7 shows the percentage of failure in each category for Site 2. The right-most bar shows the breakdown of all the failure recorded in Site 2, whilst the first three bars are for resources A, B and C respectively.

We can see that software and hardware failures are the largest contributors to failures. In the case of Site 1, the actual percentage for software ranges from 28.21% to 35.29%; the actual percentage for hardware ranges from 41.18% to 43.59%. Overall, in Site 1, the two categories are responsible for 73.55% of all the failures recorded for the site.

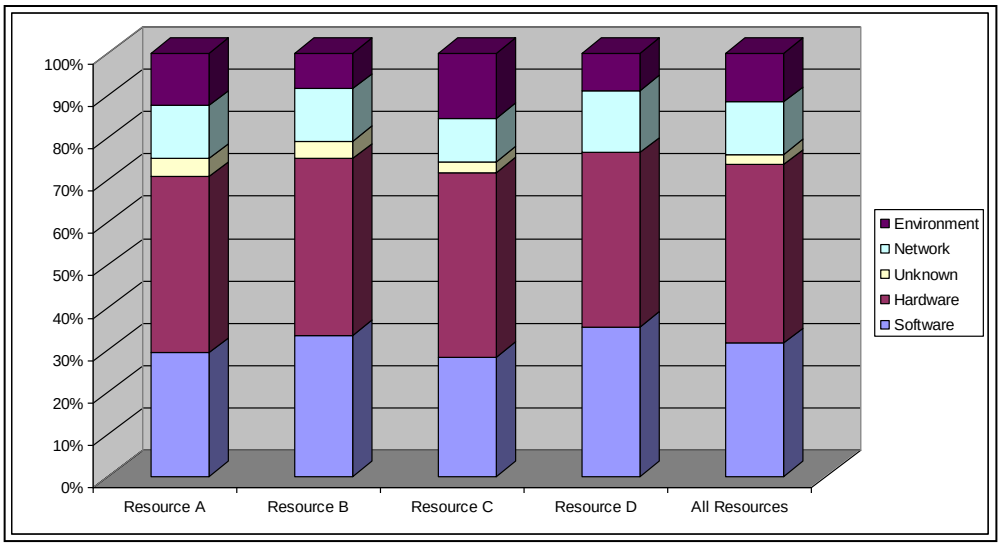


Figure 6: Breakdown of Failures into Root Causes for Resources from Site 1.

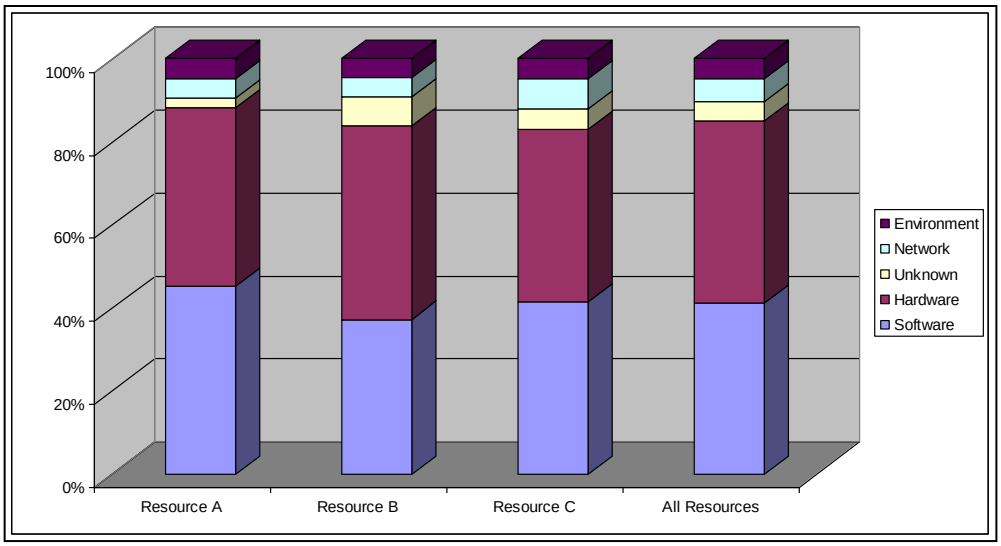


Figure 7: Breakdown of Failures into Root Causes for Resources from Site 2.

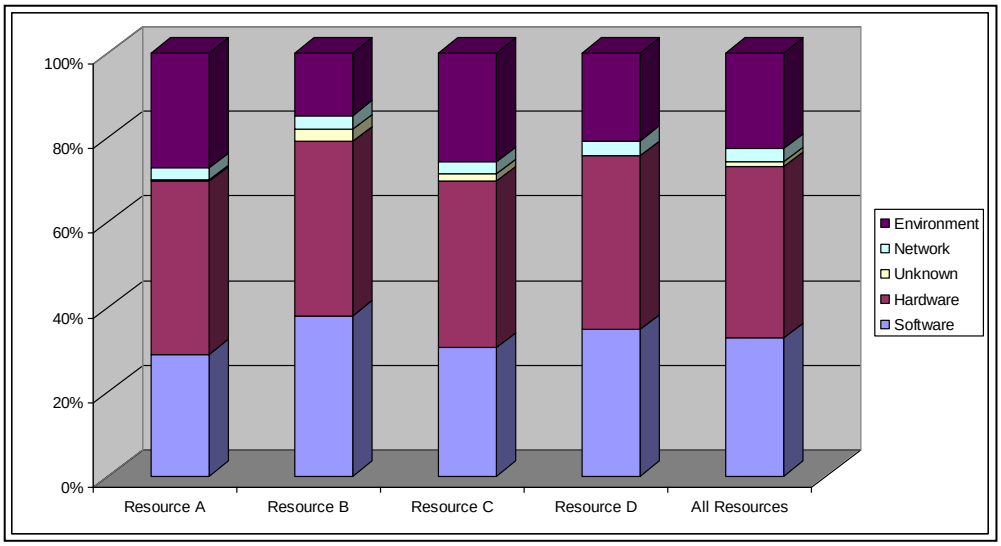


Figure 8: Breakdown of Downtime into Root Causes for Resources from Site 1.

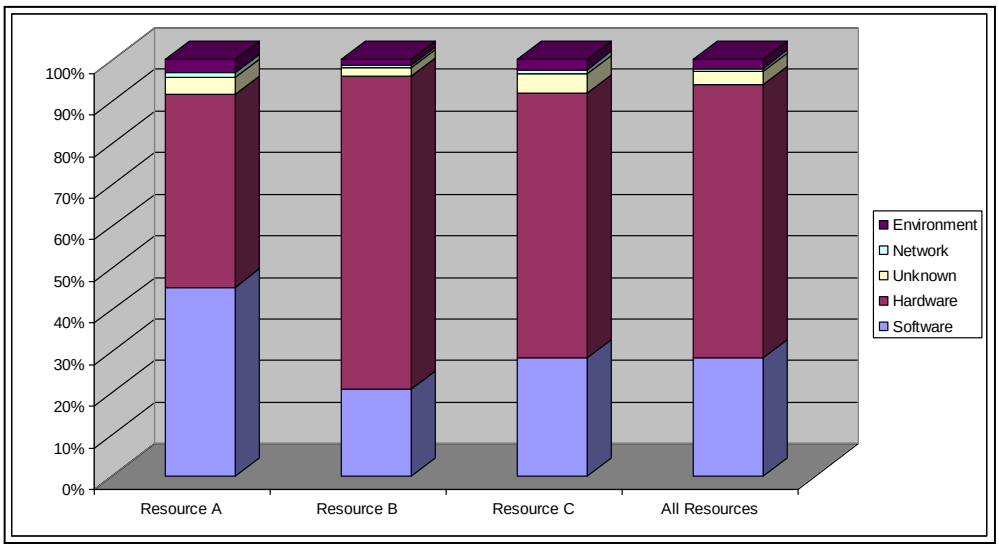


Figure 9: Breakdown of Downtime into Root Causes for Resources from Site 2.

In Site 2, the actual percentage for software ranges from 37.21% to 45.24%. The actual percentage for hardware ranges from 41.46% to 46.51%. Overall, in relation to Site 2, the two categories are responsible for 84.92% of all the failures recorded for the site.

The total downtime has been studied for each category. Figure 8 shows the percentage of downtime for each category in Site 1. The right-most bar emphasises the breakdown of all the downtime recorded in the four resources, whilst the first four bars are for resources A, B, C and D respectively.

We can see that software and hardware failures contribute hugely to the downtime. Downtimes owing to software failures contribute from 28.82% to 37.86%, whilst downtimes due to hardware failures contribute from 39.26% to 41.14%. Overall, in Site 1, the two categories are responsible for 73.14% of all the downtimes recorded in the database. In Site 1, downtime due to environment failures is high, ranging from 14.80% to 27.21%; the reason for this is that the site had an air conditioning failure, which required a long maintenance work.

Figure 9 shows the percentage of downtime for each category in Site 2. The right-most bar shows the breakdown of all the downtime recorded in the three resources, whilst the first four bars are for resources A, B and C respectively.

We can see that software and hardware failures contribute hugely to the downtime. Downtimes owing to software failures contribute from 20.48% up to 45.35%, whilst downtimes due to hardware failures contribute from 46.40% to 75.27%. Overall, in Site 2, the two categories are responsible for 93.94% of all the downtimes recorded in the database.

4.3.4 Repair Time Analysis

The second important metric in studying failures is the time to repair the system. We start by considering how the repair time varies between resources. Next, the statistical proprieties of repair time for each resource are taken into account—including their distributions. Finally, how the root cause affects the repair time is taken into account.

Tables 1 & 2 show, in minutes, the mean, median and standard deviation for the time to repair resources in Site 1 and Site 2 respectively. The mean time to repair in all resources is very high, especially resources in Site 1. The first reason is that

the repair time depends hugely on the availability of the Grid administrator, and both sites do not have 24-hour user support. Thus, any resource failure occurring after normal working hours is not resolved until the next working day; this is also true for weekends and public holidays. The second reason is that there is no automatic monitoring which will report a resource failure when it occurs. Finally, both sites are mainly used for research—not commercial use. In order to improve the mean repair time, the sites should increase the availability of administrators and deploy automatic monitoring agents.

Table 1: Repair Mean Median and Standard Deviation for Resources in Site 1 in Minutes.

	Resource A	*Resource B*	*Resource C*	*Resource D*
Mean	1922.50	1611.96	1658.85	1829.35
Median	945.50	433.50	1116.00	865.00
Standard Deviation	2496.19	2341.05	2089.17	2346.35

Table 2: Repair Mean Median and Standard Deviation for Resources in Site 2 in Minutes.

	Resource A	*Resource B*	*Resource C*
Mean	397.69	868.40	537.54
Median	200.50	240	240
Standard Deviation	472.77	2179.69	917.89

Another observation is that the time to repair a resource is highly variable owing to the difference between the mean and the median. This observation indicates that the exponential distribution is not conventional to express repair time in Grid resources. With this in mind, it should be noted that an Exponential distribution with failure rate = λ the mean = $1/\lambda$ and median = $\ln(2)/\lambda = 0.6931/\lambda$ [10]; thus, the mean and median should not have a huge difference. To confirm this observation, the empirical Cumulative Distribution Function (cdf) for repair time in each resource is fitted with four standard distributions: Exponential, Weibull, Gamma and Lognormal. The cdf—referred to as *F(x)*—describes the probability distribution of a real-valued random variable X to be less than *x*.

$$F(x) = P\{X < x\}$$

That is, for a given value x, $F(x)$ is the probability that the observed value of X will be at most x.

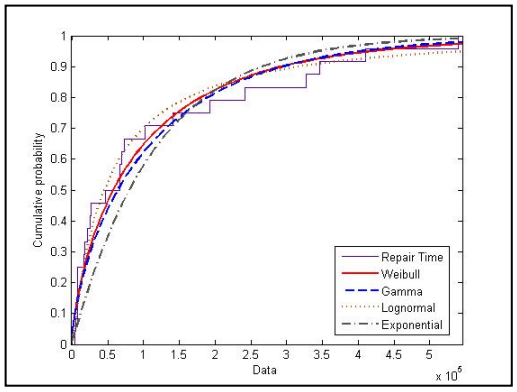


Figure 10: Repair Time Resource A Site 1.

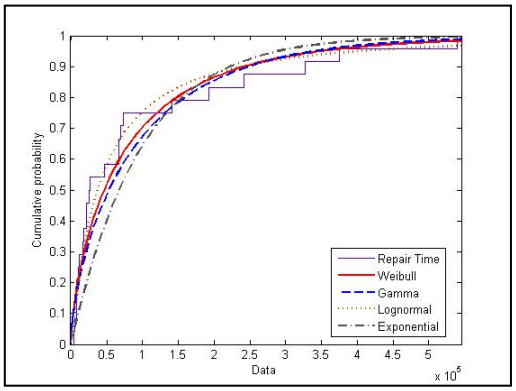


Figure 11: Repair Time Resource B Site 1.

Figure 10 shows the cdf of repair time for Resource A, Site 1. Visual inspection indicates both Lognormal and Weibull have a good fit, but that Lognormal fit the data slightly better when tested using the negative log-likelihood. The Exponential distribution is the worst fit, as expected, and it is not accurate for the purpose of modelling the repair time of this resource. The Lognormal or the Weibull is a better model for the repair time.

Figure 11 shows the cdf of repair time for Resource B, Site 1. Weibull and Lognormal distributions have a good visual fit with Weibull having the best fit when measured by the negative log-likelihood. Figure 12 shows the cdf of repair time for Resource C, Site 1. Both Weibull and Lognormal distributions have a good visual fit, yet Lognormal fit the data slightly better when tested using the negative log-likelihood. Figure 13 shows the cdf of repair time for Resource D, Site 1. Both Weibull and Lognormal distributions create an equally good visual fit, and the same negative log-likelihood.

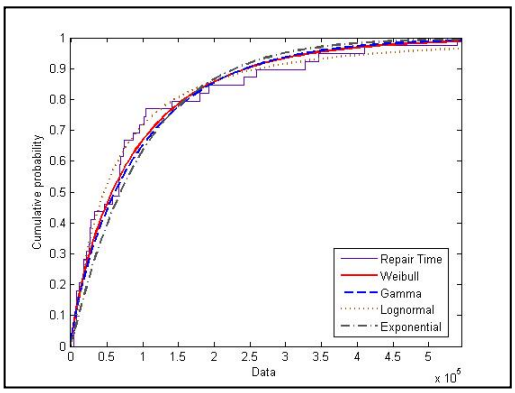


Figure 12: Repair Time Resource C Site 1.

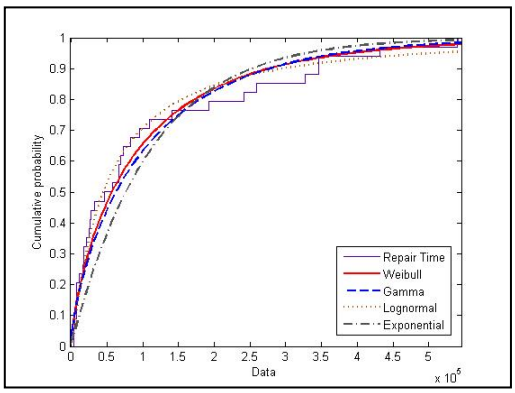


Figure 13: Repair Time Resource D Site 1.

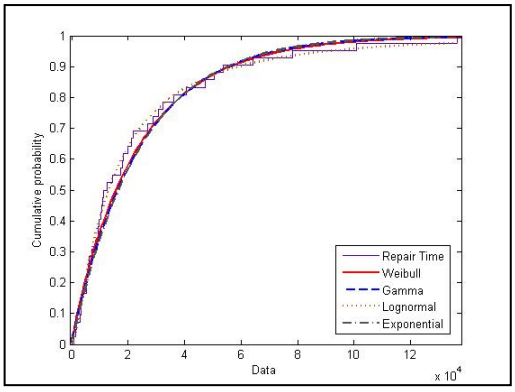


Figure 14: Repair Time Resource A Site 2.

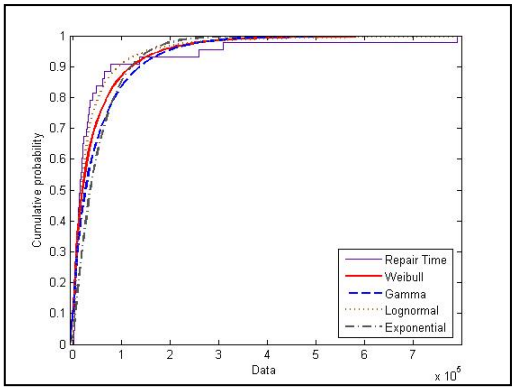


Figure 15: Repair Time Resource B Site 2.

Figure 14 shows the cdf of repair time for Resource A, Site 2. Both Weibull and Lognormal distributions create an equally good visual fit, yet Lognormal fit the data slightly better when tested using the negative log-likelihood.

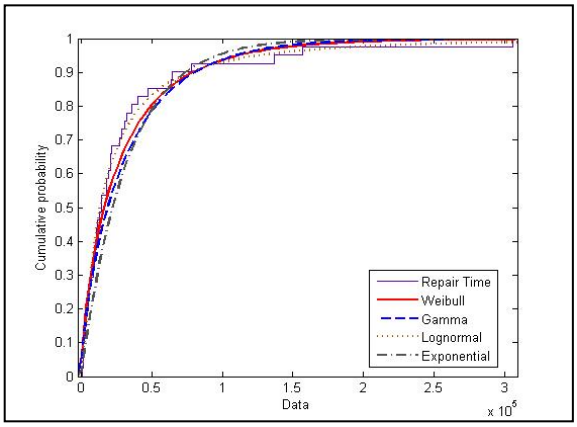


Figure 16: Repair Time Resource C Site 2.

Figure 15 shows the cdf of repair time for Resource B, Site 2. Both Weibull and Lognormal distributions have a good visual fit, with Lognormal having the best fit when measured by the negative log-likelihood.

Finally Figure 16 shows the cdf of repair time for resource C site 2. Lognormal distributions have the best visual fit and the best fit when measured by the negative log-likelihood.

From the above results, two observations can be made: firstly, it is clear that time to repair a Grid resource does not follow an Exponential distribution; and secondly, it is better to describe the repair time in the form of the Lognormal distribution, with the Weibull distribution slightly the second best.

Table 3: Mean Median and standard Deviation of Time to Repair Resource A Site 1 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	1900	1887	432	4185	1120
Median	1120	961	120	5444	1120
Standard Deviation	2136.72	2710.19	593.12	3451.25	Undefined

Table 4: Mean Median and standard Deviation of Time to Repair Resource B Site 1 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	1830.88	1589.90	432	2862.50	1120
Median	374.50	597.50	120	2862.50	1120
Standard Deviation	2360.94	2688.80	593.12	3650.79	Undefined

Now we consider how the root cause of failure affects the repair time. Tables 3, 4 , 5 & 6 show for Site 1 in minute the mean, median and standard deviation of time to repair as a function of root causes for resources A, B, C and D respectively: the mean repair time in Resource A ranges from around 7 hours in network errors to around 70 hours in environment errors; in Resource B, the mean repair time ranges from around 7 hours in network errors to around 48 hours in environment errors; in Resource C, the mean repair time ranges from around 8 hours in network errors to around 47 hours in environment errors; and finally, in Resource D, the mean repair time ranges from around 8 hours in network errors to around 72 hours in environment errors.

The second observation from Site 1 is that the time to repair is highly variable in all resources. For example, the median of network repair times is approximately 4 times lower than the mean in Resource A; the median of software repair times is about 5 times lower than the mean in Resource B; the median of hardware repair times is about 2 times lower than the mean in Resource C; and the median of hardware repair times is about 2 times lower than the mean in Resource D.

Table 5: Mean Median and standard Deviation of Time to Repair Resource C Site 1 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	1788.18	1494.18	460.75	2776.83	1120
Median	971.00	775.00	333.50	1483	1120
Standard Deviation	1994.51	2168.25	487.68	2681.96	Undefined

Table 6: Mean Median and standard Deviation of Time to Repair Resource D Site 1 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	1790	1827.86	440.60	4308.33	Null
Median	1263.50	865	360	5444	Null
Standard Deviation	1888.89	2597.92	424.74	3596.59	Undefined

In Site 1, there was only one unknown error in resources A, B and C; therefore, the standard deviation is undefined for these resources. In Resource D, there were no unknown errors, and so the mean, median and standard deviation are undefined.

Finally, in Site 1, software and hardware failure effects are on individual resources, whilst a network or an environment failure may affect more than one resource—or even the entire Grid site. For example, a power cut in the Grid site will result in the failure of all resources in that site.

Table 7: Mean Median and standard Deviation of Time to Repair Resource A Site 2 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	398.63	430.54	92	260	675
Median	303.00	157.90	92	260	675
Standard Deviation	323.95	636.13	98.99	98.99	Undefined

For Site 2, Tables 7, 8 & 9 show in minutes the mean, median and standard deviation the time to repair as a function of root causes for resources A, B and C respectively: the mean repair time in Resource A ranges from around 1.5 hours in network errors to around 7 hours in hardware errors; in Resource B, the mean repair time ranges from around 1.5 hours in network errors to around 23 hours in

environment errors; and finally, in Resource C, the mean repair time ranges from around 1 hour in network errors to around 14 hours in environment errors.

The second observation from Site 2 is, like Site 1, the time to repair, which is highly variable in all resources. For example, the median of hardware repair times is about 3 times lower than the mean in Resource A; the median of hardware repair times is about 6 times lower than the mean in Resource B; and the median of hardware repair times is about 5 times lower than the mean in Resource C.

In Site 2, there was only one unknown error in Resource A; therefore, the standard deviation is undefined for the resource.

Table 8: Mean Median and standard Deviation of Time to Repair Resource B Site 2 Breakdown by Root Causes in Minutes.

	Software	*Hardware*	*Network*	*Environment*	*Unknown*
Mean	479.25	1385.05	92	260	364
Median	302	220	92	260	364
Standard Deviation	576.14	3112.18	98.99	98.99	439.82

Table 9: Mean Median and standard Deviation of Time to Repair Resource C Site 2 Breakdown by Root Causes in Minutes.

	Software	Hardware	Network	Environment	Unknown
Mean	366.53	826.06	73	260	513
Median	301	158	35	260	513
Standard Deviation	312.78	1354.32	77.35	98.99	229.10

Finally, like in Site 1, software and hardware failures in Site 2 effects are on individual resources, whilst a network or an environment failure may affect more than one resource—or even the entire Grid site.

4.3.5 Time between Failures Analysis

In this section, the sequence of failure events are viewed as a stochastic process, and we study the time between unscheduled failures, inter-arrival times, for each resource. The cdf for the time between failures in each resource is fitted with four standard distributions: Exponential, Weibull, Gamma and Lognormal.

Figures 17, 18, 19 & 20 show, for Site 1, the cdf of time between failures for resources A, B, C and D respectively. In the case of Resource A, the distribution between failures is well modelled by a Weibull distribution, which creates a good

visual fit and the best fit when tested using the negative log-likelihood. The Gamma distribution is the second best fit.

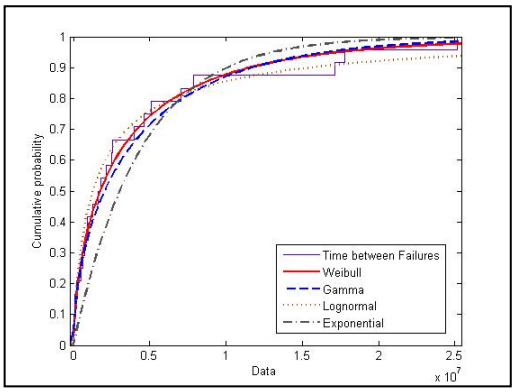


Figure 17: Time between Failures for Resource A Site 1.

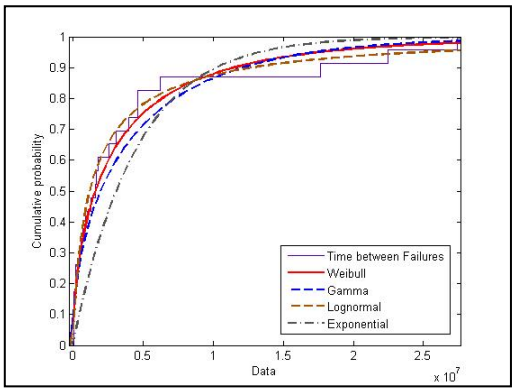


Figure 18: Time between Failures for Resource B Site 1.

In Resource B, the distribution between failures is well modelled by a Weibull distribution, which creates a good visual fit and the best fit when tested using the negative log-likelihood. The Gamma or the Lognormal distributions are the second best fit.

In Resource C, the distribution between failures is well modelled by a Weibull distribution, which creates a good visual fit and the best fit when tested using the negative log-likelihood. The Gamma distribution is the second best fit.

Finally, in Resource D, the distribution between failures is well modelled by a Weibull or Gamma distribution. Both distributions create an equally good visual fit and the same negative log-likelihood.

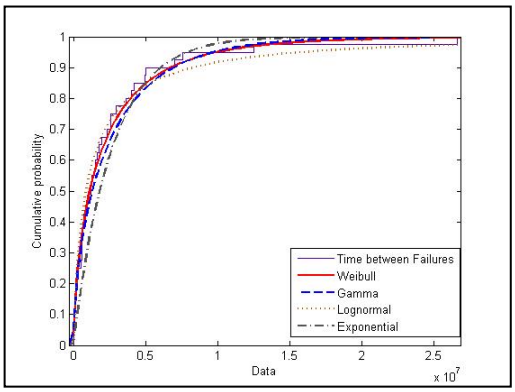


Figure 19: Time between Failures for Resource C Site 1.

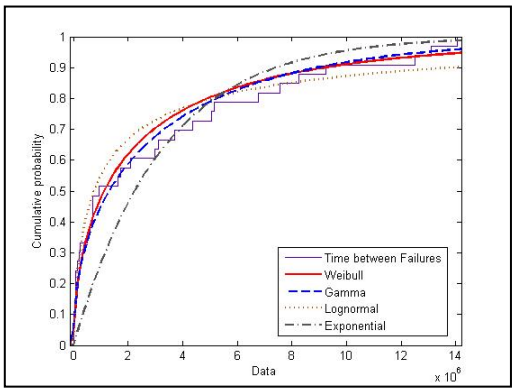


Figure 20: Time between Failures for Resource D Site 1.

For Site 2, Figures 21, 22 & 23 show the cdf of time between failures for resources A, B and C respectively.

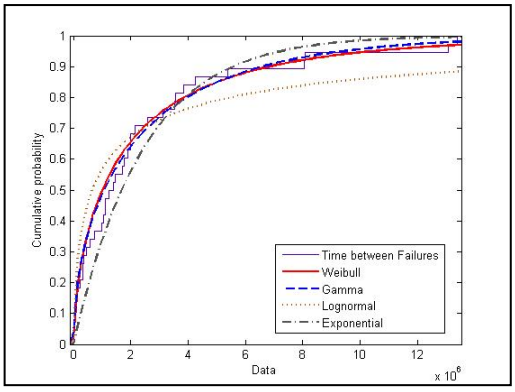


Figure 21: Time between Failures for Resource A Site 2.

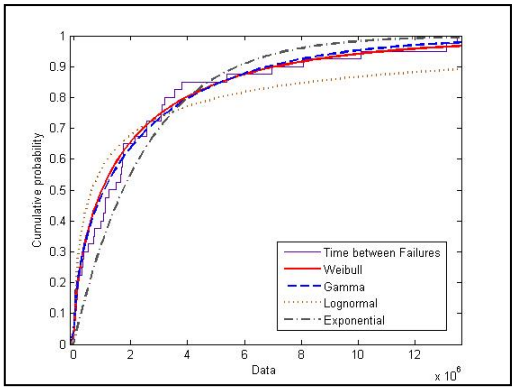


Figure 22: Time between Failures for Resource B Site 2.

In Resource A, the distribution between failures is well modelled by a Weibull or Gamma distribution. Both distributions create an equally good visual fit and the same negative log-likelihood.

In Resource B, the distribution between failures is well modelled by a Weibull distribution, which creates a good visual fit and the best fit when tested using the negative log-likelihood. The Gamma distribution is the second best fit.

Finally, in Resource C, the distribution between failures is well modelled by a Weibull or Gamma distribution. Both distributions create an equally good visual fit and the same negative log-likelihood.

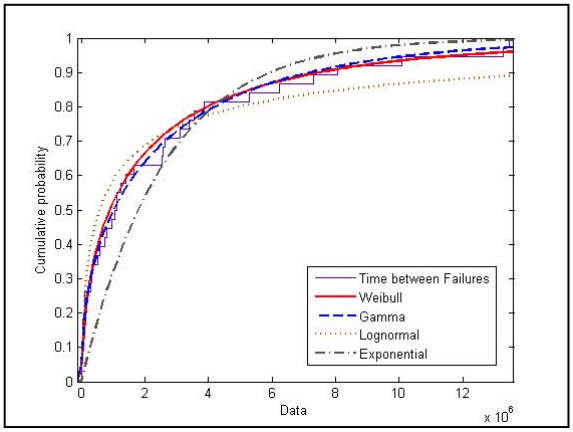


Figure 23: Time between Failures for Resource C Site 2.

From the above, we can state that the Weibull distribution is the best distribution to model distribution between failures in Grid resources where as the Gamma distribution is the second best fit. The Weibull distribution is the most popular and widely used method of analysing and predicting failures and malfunctions of all types, offers flexibility in modelling failure rates, and is easy to calculate [197-200].

The Weibull distribution mathematically characterizes the probability of system failures as a function of time. The two parameters Weibull function is used in this thesis and the probability density function *pdf* is defined as:

$$f(t) = \frac{\alpha}{\lambda}\left(\frac{t}{\lambda}\right)^{\alpha-1} e^{-\left(\frac{t}{\lambda}\right)^{\alpha}}$$

The cumulative density function *cdf* is defined as:

$$F(t) = 1 - e^{-\left(\frac{t}{\lambda}\right)^{\alpha}}$$

Where α is the shape parameter (or slop), λ is the scale parameter and t is time. Recalling that the reliability function of a distribution is simply one minus the *cdf*, the reliability function for the Weibull distribution is given by:

$$R(t) = 1 - F(t)$$

From the above, we can calculate the Weibull failure rate (or hazard rate) function as follow:

$$h(t) = \frac{f(t)}{R(t)} = \alpha t^{\alpha-1} \lambda^{-\alpha}$$

The shape parameter α directly influences the hazard function as follows:

If α < 1, the hazard function is decreasing with time;

If α = 1, the hazard function is constant with time, i.e., the exponential distribution;

If α > 1, the hazard function is increasing with time.

It is useful to determine how the time since the last failure influences the expected time until the next failure; this notion is captured by a distribution's hazard rate function. An increasing hazard rate function predicts that the probability of failure increases with time. A decreasing hazard rate function predicts the reverse. The shape parameter of less than 1 indicates that the hazard rate function is decreasing, i.e. not seeing a failure for a long time decreases the chance of seeing one in the near future.

In this thesis, we use the maximum likelihood estimation to predict the parameters and we find decreasing hazard rates a Weibull shape parameter less than 1; this means not seeing a failure for long time decreases the risk of seeing one within a short period of time. Table 10 shows the values of the Weibull shape parameter for the resources.

Table 10: The Weibull Shape Parameter.

Site One	*Resource A*	*Resource B*	*Resource C*	*Resource D*
	0.63618	0.609953	0.673741	0.569431
Site Two	*Resource A*	*Resource B*	*Resource C*	
	0.623174	0.607578	0.564124	

4.4 Probabilistic Failure Models for Grid Resources

The Weibull failure rate function provides the probability of resource failure up to a point in time, without considering what happens if the resource fails during that time and is then repaired. Grid resources are repairable systems and receive maintenance actions when they fail. The maintenance actions might change the overall makeup of the resource, and must be taken into consideration when assessing the probability of failure of the resource as the age of the resource components is no longer identical and the time of operation is not continuous.

In the previous sections, the focus has been directed onto describing the behaviour of Grid resources in statistical terms. The distribution failure rate functions focus on the first time to failure, or first time to failure in a given interval—but not whether the resource is functioning or not functioning at a given time. The resource availability function capture the notation of resource functioning [177]. *Point availability* at time *t* is the probability of the resource functioning at time *t* and is denoted by *A(t)*. The average proportion a resource is functioning during an interval (t_1, t_2) is denoted by $A_v(t_1, t_2)$, and can be obtained by the following formula:

$$A_v(t_1, t_2) = \frac{\int_{t_1}^{t_2} A(u)du}{t_2 - t_1}$$

In order to compute the Grid resource availability, a model for the resource needs to be driven. Models from reliability engineering can be used to represent a Grid resource and to thereby predict the probability of failure. The problem in this regard is which model to use. Random processes are widely used as probabilistic models for the failure process [9]; the following is a list of random processes and a discussion on their ability to model Grid resources failure.

- **Renewal Process and Homogeneous Poisson Process:** A renewal process assumes that, upon failure, the system is instantaneously repaired to an 'as good as new' state. It also assumes that the distribution of the time between failures is identical and independent (IID). The homogeneous Poisson process (HPP) is a special case of the renewal process, in which the time between failures follows the exponential distribution. Grid resources cannot be modelled as HPP as the distribution of the time between failures for these resources is Weibull and not exponential (see 4.3.5). Furthermore, Grid resources cannot be modelled as a renewal process for two reasons: first, the repair of a resource will not return it to an 'as good as new' state. Second, a resource changes during repairs and assuming identical distribution is inadequate.

- **Modified Renewal Process:** A modified renewal process is a process with the distribution of the first failure differs from the distribution of the time of the second, third or subsequent failures. Grid resources cannot be modelled as a modified renewal process as the distribution of the time between failures does not change between subsequent failures.

- **Alternating Renewal Process:** An alternating renewal process does not assume an instantaneous repair, and takes into account the time to repair a failed system. Grid resources cannot be modelled as an alternating renewal process as the alternating renewal process assumes an IID failures and Grid resources change during repairs.

- **Non-Homogeneous Poisson Process:** A non- homogeneous Poisson process (NHPP) is an extension on HPP whereby the rate of failure, as given by the rate of occurrence of failures (ROCOF), is assumed to vary with time. The ROCOF function is also referred to as the *peril rate*. The NHPP is widely assumed in modelling computer systems as the rate of failures varies with time and the distribution of the time between failures is not assumed to be identical. Two NHPP models are widely used in reliability engineering: the NHPP following a Power Low and the NHPP Following an Exponential Low [196].

4.4.1 NHPP Following a Power Low

The power low model—also known as Crow's model or Weibull process—because the time to the first failure has a Weibull distribution, has the following ROCOF [201]:

$$h(t) = \lambda\beta t^{\beta-1}, \qquad \lambda, \beta > 0, \qquad t \geq 0$$

where λ is the scale parameter, β is the growth parameter and t is the time.

In 1964, Duane [202] introduced the technique of plotting the cumulative failure rate against t on a log-log paper. If the system generating the failures follows a power-low model then, subject to sampling variability, a liner plot will be obtained on the log-log paper. The cumulative failure rate is $N(t_i)/t_i$ where $N(t)$ is a counting function which keeps track of the cumulative number of failures the system has had from time zero to time t, where t_i is the time of the i_{th} failure.

Figures 24, 25, 26, 27, 28, 29 & 30 show the Duane plots of failures for resources A, B, C, D from Site 1, A, B and C from Site 2 respectively. The X Axis represents $ln(t)$ and the Y Axis represents $ln(t/N(t))$[1]. From the figures, it can be seen that the points in the plots are scattered and do not form a roughly linear plot. Therefore, Grid resources, most likely, cannot be modelled as a power low NHPP. Furthermore, the resources repair time is not modelled as a power-low NHPP (see Appendix A for the Duane plots of repair time).

[1] The implementation of Duane plot that's put $t_i/N(t_i)$ on the vertical axis is used.

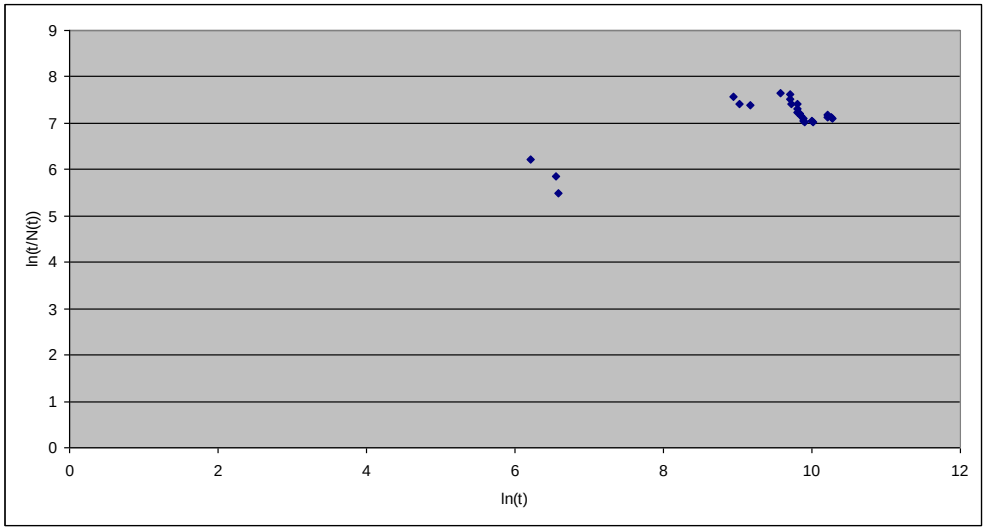


Figure 24: The Dune Plot for Failures of Resource A Site 1.

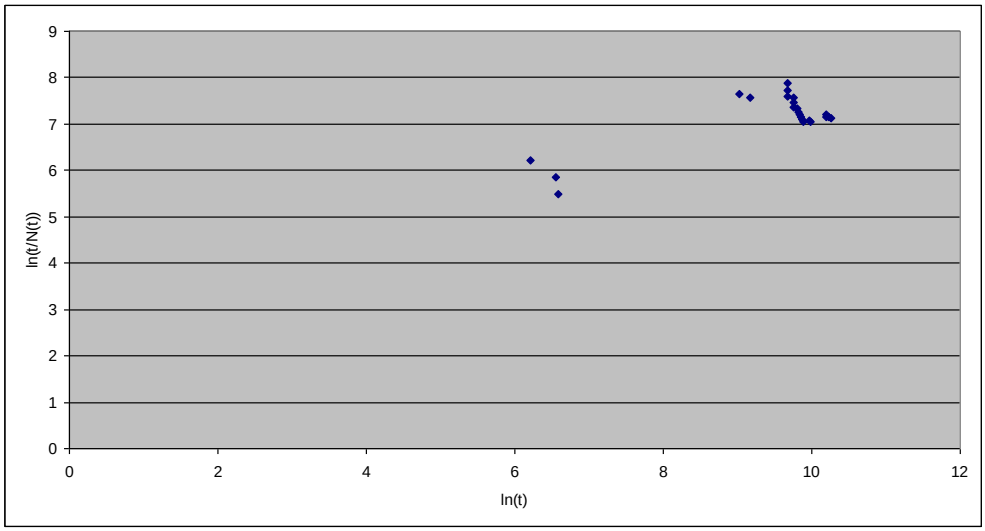


Figure 25: The Dune Plot for Failures of Resource B Site 1.

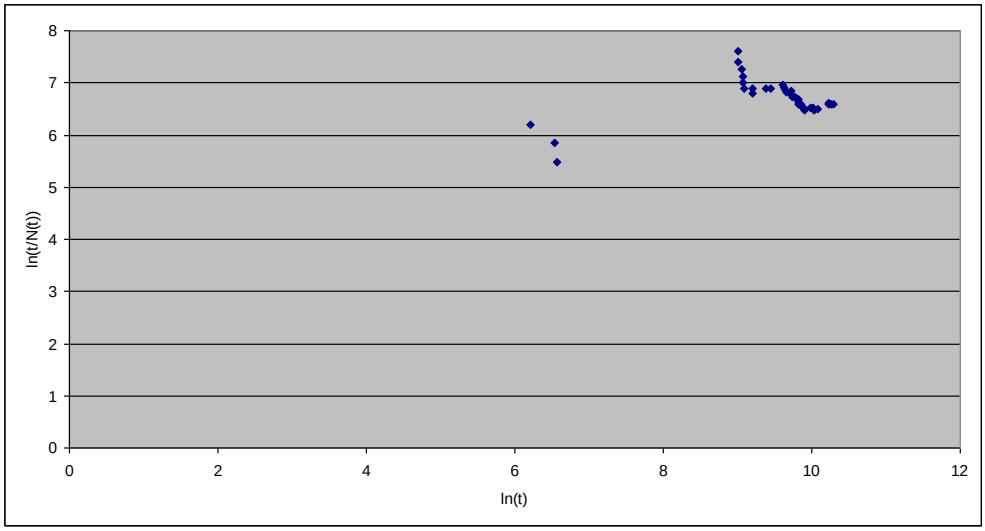


Figure 26: The Dune Plot for Failures of Resource C Site 1.

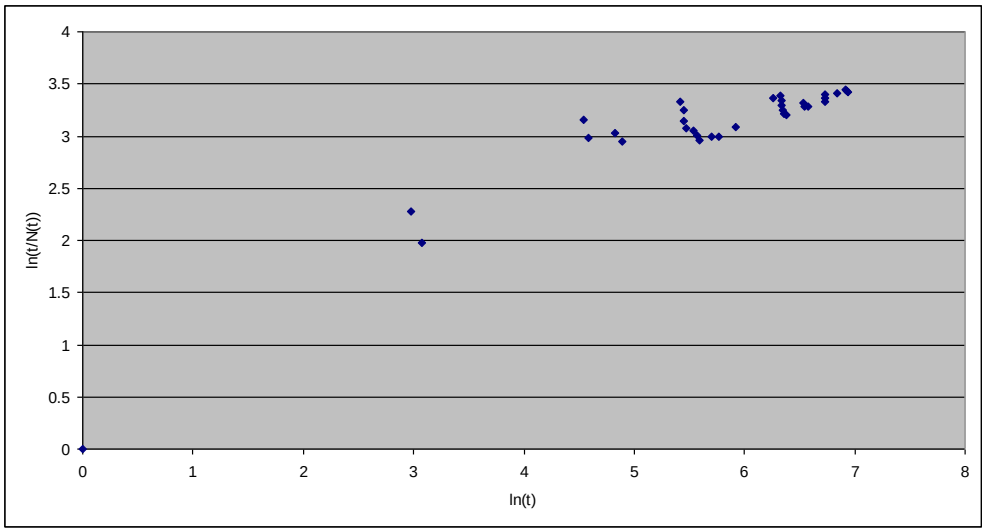


Figure 27: The Dune Plot for Failures of Resource D Site 1.

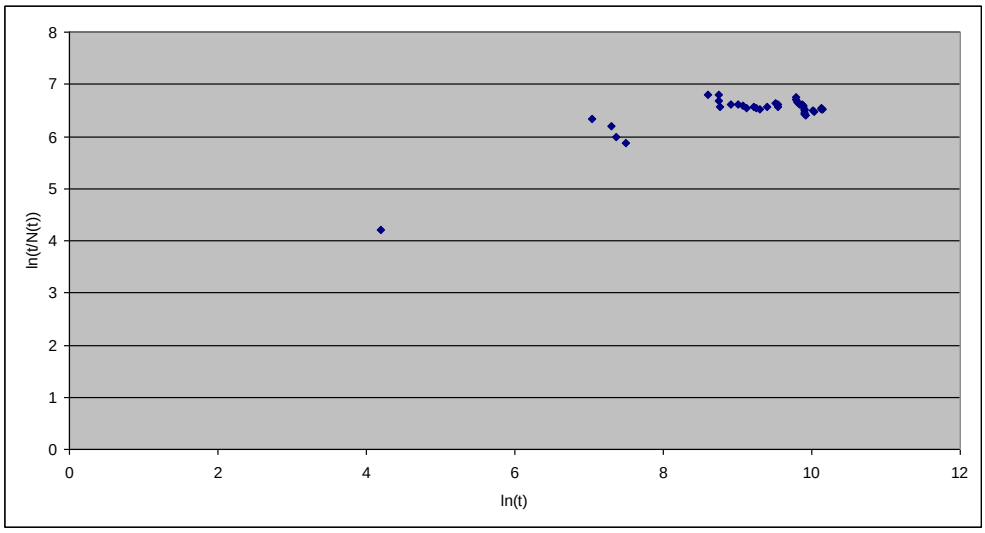


Figure 28: The Dune Plot for Failures of Resource A Site 2.

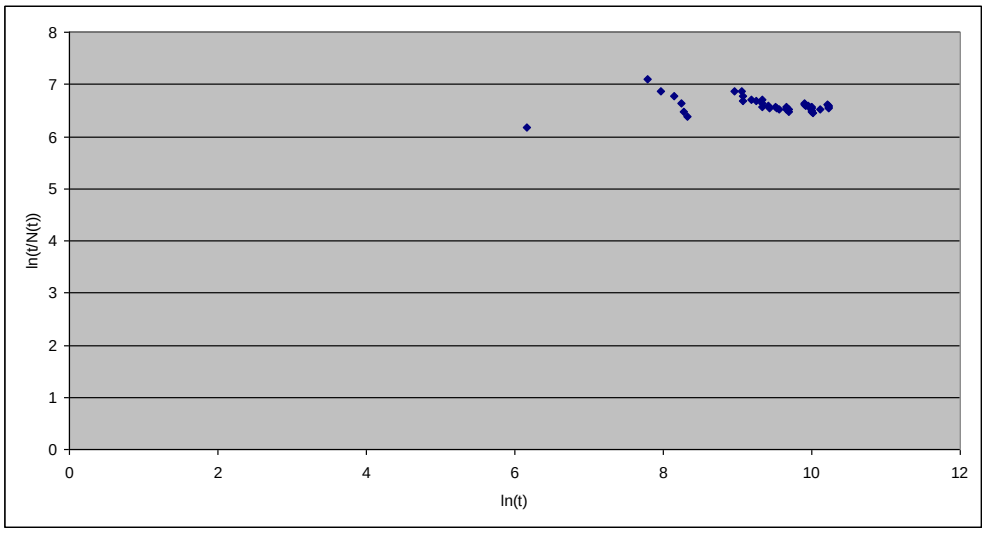


Figure 29: The Dune Plot for Failures of Resource B Site 2.

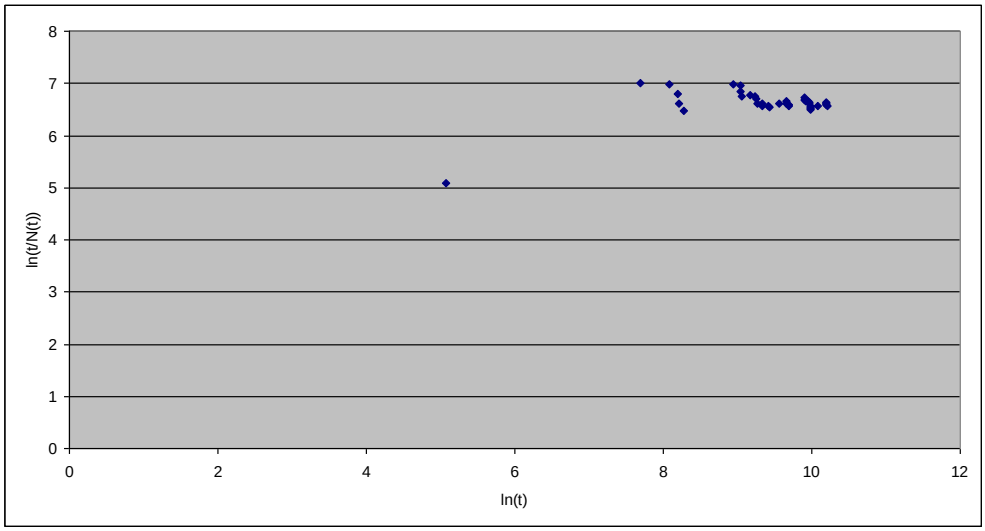


Figure 30: The Dune Plot for Failures of Resource C Site 2.

4.4.2 NHPP Following an Exponential Low

The exponential low model—also known as Cox and Lewis's model—has the following ROCOF [196]:

$$h(t) = e^{a_0 + a_1 t}$$

where a_0 is the scale parameter, a_1 is the growth parameter and t is the time.

Plotting the cumulative failure rate against t on a log-linear paper should roughly follow a straight line if the system generating the failures follows an exponential low NHPP.

Figures 31, 32, 33, 34, 35, 36 & 37 show the plots of the cumulative failure rate against t on a log-linear paper for resources A, B, C, D from Site 1, A, B and C from Site 2 respectively. The X Axis represents the time t in hours, whilst the Y Axis represents $ln(t/N(t))$. The figures show that the points on the plots do not form a roughly linear plot; therefore, Grid resources cannot be modelled as an exponential low NHPP. Moreover, the resources repair time is not modelled as an exponential low NHPP (see Appendix A for the plot of the cumulative repair rate against t on a log-linear paper).

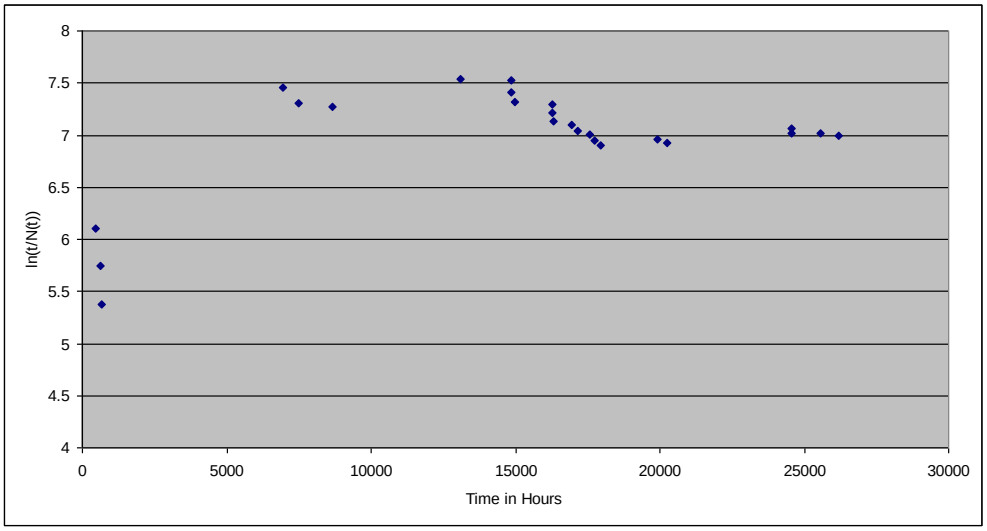


Figure 31: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource A Site 1.

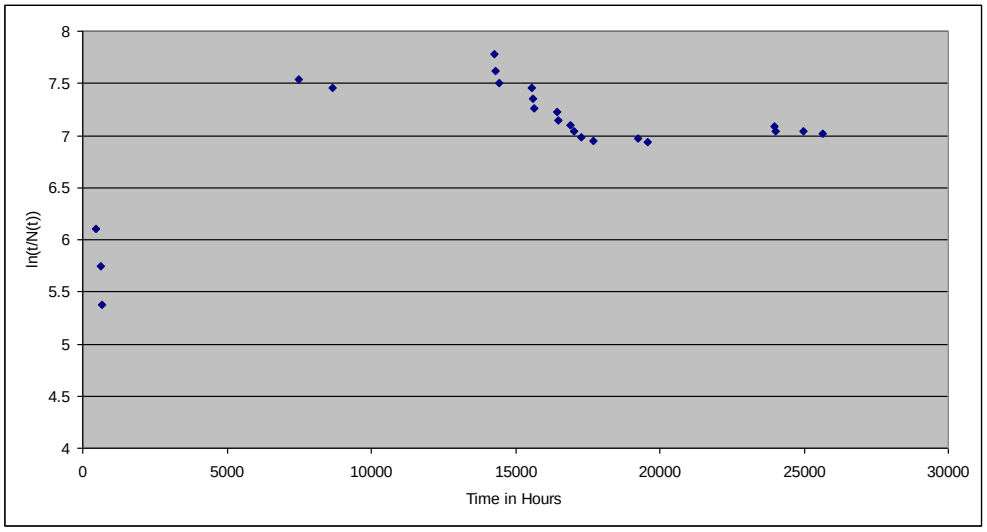


Figure 32: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource B Site 1.

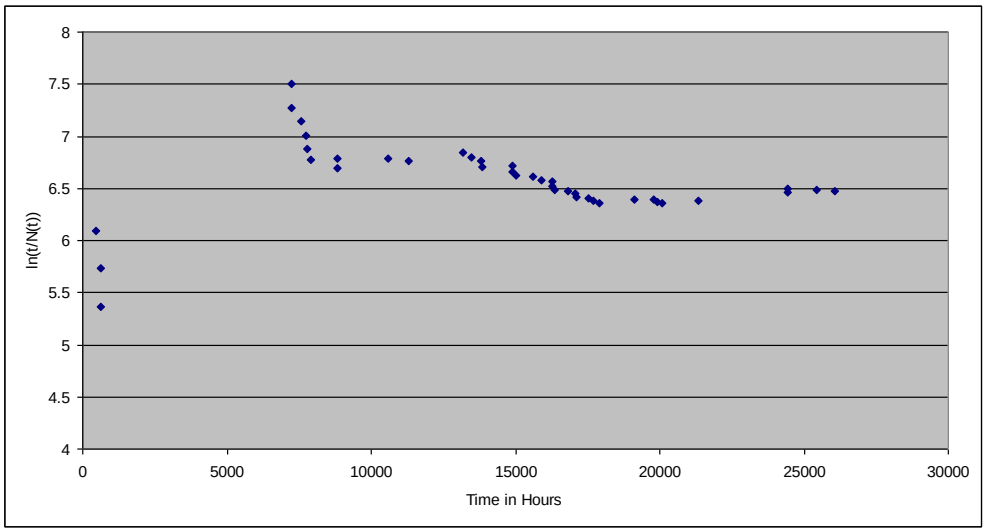


Figure 33: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource C Site 1.

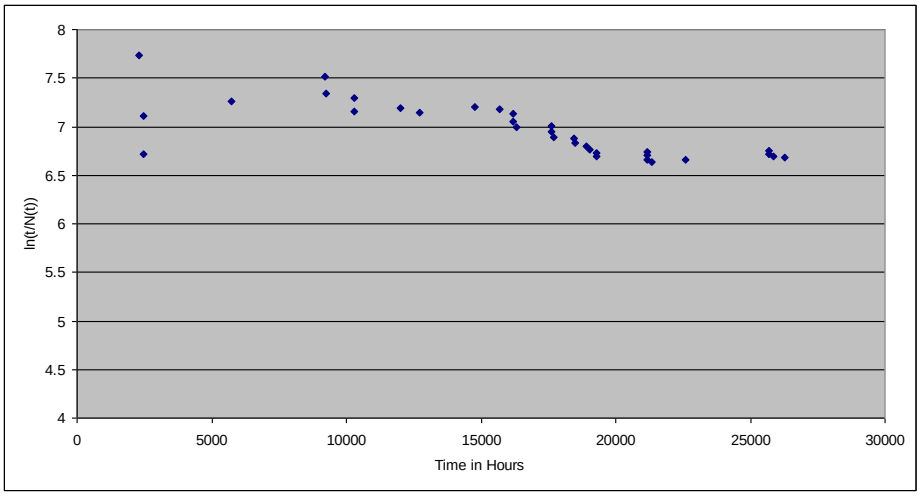


Figure 34: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource D Site 1.

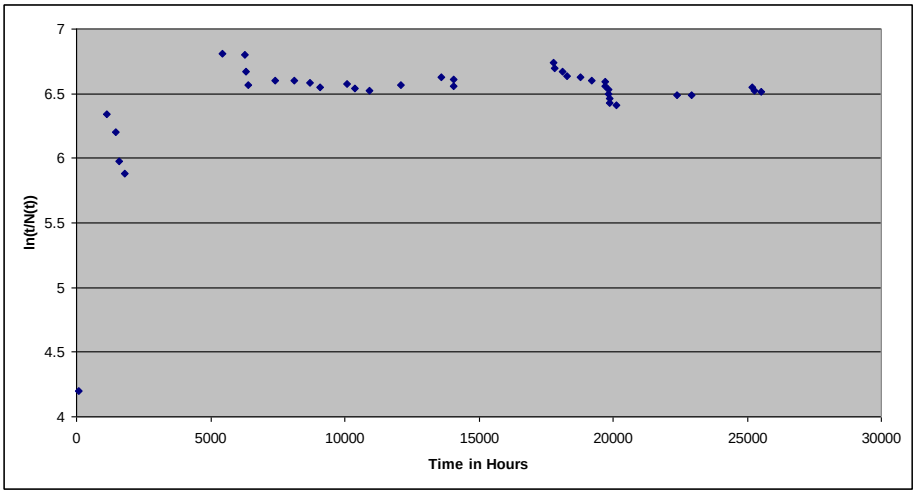


Figure 35: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource A Site 2.

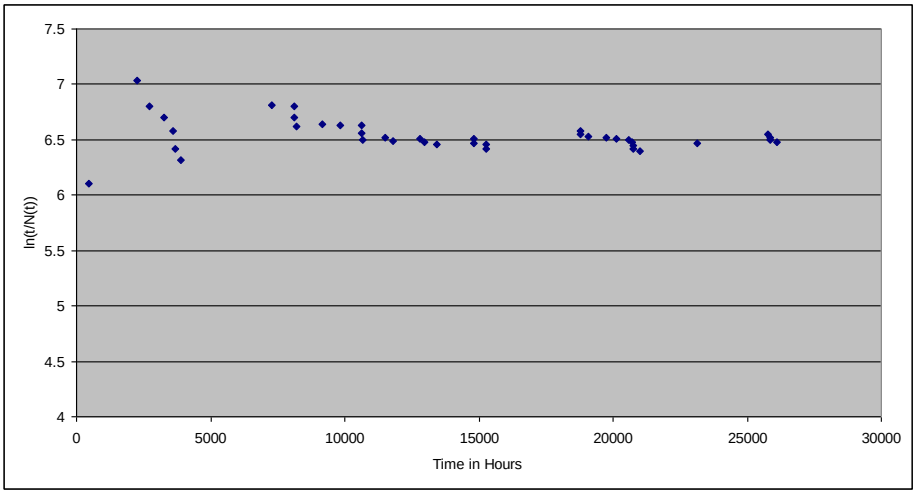


Figure 36: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource B Site 2.

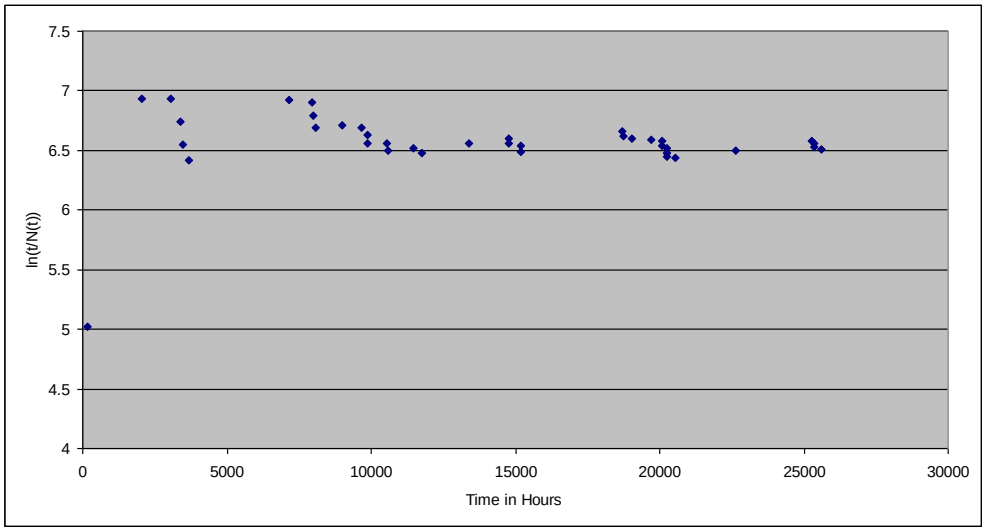


Figure 37: Cumulative Failure Rate against *t* on a log-linear Paper for Failures of Resource C Site 2.

4.4.3 Results Analysis

In this section the random processes are tested as probabilistic models for the Grid resources failure. The results show that random processes are not suitable for modelling Grid resources failure. The HPP assumes that the time between failures follows the exponential distribution, yet the time between failures in grid environments follows a Weibull distribution. The renewal process assumes that the repair of failed component return it to "as good as new" state, yet in Grid environments repairs do not return the resources to as good as new state. The modified renewal process assumes that the distribution of the first failure differs from the distribution of the time of the second, third or subsequent failures. This assumption is not valid in Grid environments since the distribution of the time between failures follows the Weibull distribution and does not change between subsequent failures. The alternating renewal process assumes that the distribution of the time between failures is identical and independent. In Grid environments a resource changes during repairs, thus assuming identical distribution is inadequate. Finally the NHPP, which is widely assumed in modelling computer systems, is not fitting for modelling Grid resources failure. From the Dune plot it is highly unlikely that Grid resource failures are modelled by a NHPP following a power low. The same conclusion for the NHPP following an exponential low is driven from the log-linear plot.

4.5 Summary

The motivation scenario is used to demonstrate the need for risk assessment methods in order to improve the commercial uptake of Grid computing. The events that cause resource failures are identified as hardware failures, software failures, network failures, environment failures and unknown failures. The risk of failure measure is presented to be the resource probability of failure. Analysing failures records for seven different resources shows that software and hardware failures are the largest contributors to failures: the actual percentage for software ranges from 28.21% to 45.24%; the actual percentage for hardware ranges from 41.18% to 46.51%. Importantly, software and hardware failures contribute hugely to the downtime. Another observation is that the mean time to repair—in all resources—is very high, and that the time to repair a resource is highly variable owing to the difference between the mean and the median. The mean repair times vary widely depending on the root cause, and are extremely variable. The time to repair a resource is well-fitted by a lognormal distribution, with Weibull distribution as the second best. The time between failures are best fitted with a Weibull distribution with decreasing hazard rate. Finally, the assumption that failures in Grid resources are modelled by a NHPP is invalid, and the Duane plot—along with the log-linear plot—confirms this. In the next chapter, new models to estimate the resources risk of failures are introduced.

Chapter 5

Modelling Risk of Failure in Grid Environments

In the previous chapter, it has been highlighted that the Grid resource failures cannot be modelled using probabilistic failure models, such as HPP or NHPP. More advanced modelling techniques are required. These techniques are based on availability models—also known as reliability models for non-repairable systems. With this in mind, this chapter introduces a mathematical model for the prediction of the risk of failure of a Grid resource with the use of a discrete-time analytical model driven by availability functions fitted to observed data. Moreover, the model selected and the reasons for selection are presented. In addition, the different distribution of the failure data are analysed and, based on these, the risk assessment model is developed. The model is validated by comparing the proposed ROF generated by the model with the observed ROF. Finally, the use of the model to rank resources and plan future investments is studied.

5.1 Availability Models

Recall that the resource ROF at time t is the probability of the resource not functioning at time t. This can be defined as one minus the probability of the resource functioning at t. By computing the probability of the resource functioning at t, known as availability $A(t)$, the resource ROF becomes:

$$\text{ROF}(t) = 1 - A(t)$$

An availability model is an abstract mathematical and graphical representation of the system availability characteristics. The model can be evaluated so as to obtain a prediction of the system availability at a given time [203]. The taxonomy of modelling techniques for system reliability and availability is found in [177]. Two techniques are widely applied for availability: Combinatorial Models and Markov Models.

Combinatorial modelling is an approach in which the system is divided into overlapping modules. Each module is assigned a probability of working P_l, the goal of which is to drive the probability of the correct system operation. Combinatorial models have various limitations owing to the fact that they cannot be used to model system repairs or dynamic reconfiguration of the system; hence, some non-standard extensions have been added to the models so as to increase their expressiveness. Furthermore, Combinatorial Models include Reliability Block Diagram (RBD) Models, Network Models and Fault Tree Models. A serious limitation of these models is that they can only represent two states per module, i.e. operational and failed [203]. Regardless of their limitations, however, Combinatorial Models are used when the system under study is divided into modules, yet in this thesis, the Grid resource is modelled as a black-box; therefore, Combinatorial Models are not applied in this work.

Markov Models address the limitations associated with Combinatorial Models. The two central concepts of Markov models are 'state' and 'state transitions'. Recall that, from the data collected in GOCDB, Grid resources have three states.

- 'Up' the resource is fully functional, represented as State 0;
- 'At Risk' the resource will probably be working as normal, but may experience problems, represented as State 1; and
- 'Outage' the resources will be completely unavailable, represented as State 2.

As time passes, the resource moves from state to state as a result of failures and repairs. These changes in-state are known as state transitions. Markov models can be further divided into two categories: discrete-time and continuous-time models. The former, discrete-time models, require all state transitions to occur at fixed intervals, with each possible transition assigned a probability. Continuous-time models allow state transitions to occur at varying intervals, and each possible transition is assigned with a transition rate. Markov models are represented in graphs, and the information expressed by the model graph is often summarised in a square matrix **P**, whose elements $\mathbf{P}_{i,j}$ are the probability of a transition from state *i* to state *j*. The probabilistic character of the matrix requires that all elements of the matrix are non-negative, and that each row of the matrix sums one.

The basic assumption in the case of Markov models is that the resource has no memory, which implies that the transition probabilities between states are determined only by the present state and not by the history. For continuous-time models, the length of time already spent in a state does not influence either the transition rate of the next state or the remaining time in the same state before the next transition. This general assumption implies that the waiting time spent in any state is exponentially distributed in the continuous-time case or geometrically distributed in the discrete-time case. Thus, Markov models assume that failure rates are constant, thereby leading to exponentially distributed inter-arrival time of failures and Poisson arrival of failures [204]. A useful generalisation of Markov Models is the Time-Varying Markov Models, which allow state transition probability to change over time; thus, the failure rate is no longer assumed as constant [177]. With this relaxed assumption, the Grid resources can be modelled with the use of the time-varying Markov model. Since Grid resources failures and repairs occur at varying intervals, a continuous time-varying Markov model is used for Grid resource availability (see Figure 38). The transition matrix for the continuous time-varying Markov model is:

$$\boldsymbol{P}(n) = \begin{bmatrix} 0 & Z_W(t) & Z_F(t) \\ Z_R(t) & 0 & Z_F(t) \\ Z_G(t) & 0 & 0 \end{bmatrix}$$

The resource will start at State 0 and operate until either: (i) the performance degrades and the resource transits to State 1; or (ii) the resource stops working and transits to State 2. Importantly, $Z_W(t)$ is the rate of events that causes a resource to transition from State 0 to State 1, whilst $Z_R(t)$ is the rate of recovery events that result in the resource returning to State 0. Moreover, $Z_F(t)$ is the rate of events that leads to resource failure, whereas $Z_G(t)$ is the rate of repair events resulting in the resource returning to State 0.

In order to predict the Grid resource availability, the continuous time-varying Markov model is developed by applying transition functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ derived from the distributions fitted to failure data. Therefore, Section 5.2 deals with establishing distributions for the transition functions, whilst Section 5.3 presents the analysis of the model.

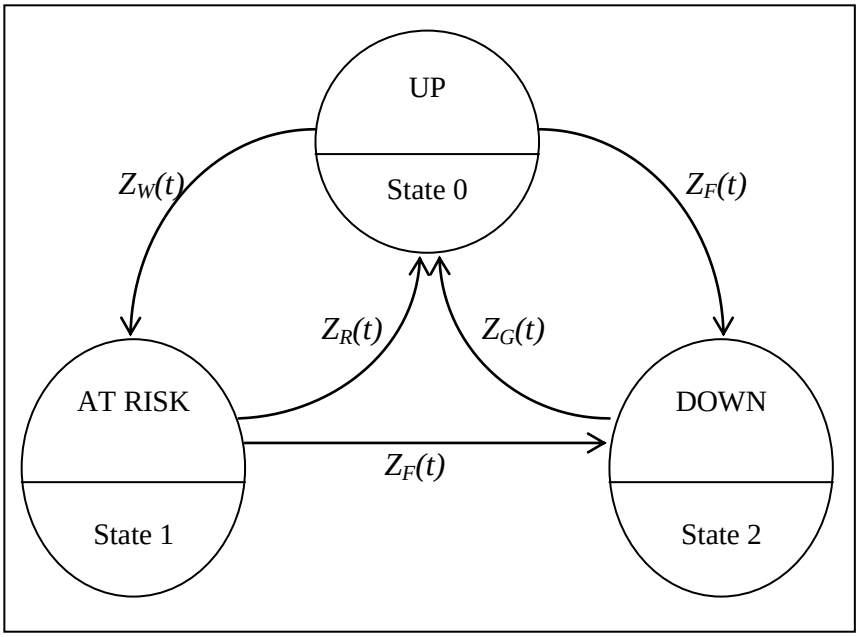


Figure 38: Continuous Time-Varying Markov Model for Resource Availability.

5.2 Fitting Distributions to Failure Data

Recall that, in Chapter 4, the downtime data for seven Grid resources from two different Grid sites are collected. Four resources are from Site 1, whilst three resources are from Site 2. We name Site 1 resources A, B, C, and D, whilst Site 2 resources are A, B and C. The downtime data for these resources will be used to drive the transition functions.

In order to determine the time-varying functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for the continuous time-varying Markov model shown above, the sequence of unscheduled events are analysed for each resource. There are two types of events: the first is At Risk, which represents a transition from State 0 to State 1; the second is complete failure, which represents the transition from State 0 to State 2. For each event, the time to repair the resource is recorded and represents the time to return the resource to State 0 from State 1 or 2. Each resource has four functions to be modelled: $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$.

1. The time between transition from ‘UP’ to ‘AT RISK’ or State 0 to State 1 denoted as $Z_W(t)$.
2. The time between transition from ‘AT RISK’ to ‘UP’ or State 1 to State 0 denoted as $Z_R(t)$.

3. The time between transition from ‘UP’ to ‘DOWN’ or State 0 to State 2 denoted as $Z_F(t)$.
4. The time between transition from ‘DOWN’ to ‘UP’ or State 2 to State 1 denoted as $Z_G(t)$.

The cdf of the four functions for each resource is fitted with four standard distributions: Exponential, Weibull, Gamma and Lognormal; this helps to determine the best fit for each function. The MLE is used to parameterise the distributions, and the goodness of fit is evaluated by visual inspection and the negative log-likelihood test.

5.2.1 Summary of Results

Resource A Site One:

From Figure 39, visual inspection shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Lognormal distribution, yet the Weibull is a better fit when tested with the use of a negative log-likelihood. The time between transitions from State 0 to 2, $Z_F(t)$, is well modelled by Weibull or Gamma; both distributions create an equally good visual fit and the same negative log-likelihood. The repair time is the time to return the resource to State 0 from State 1 or State 2. Moreover, the time between the transitions from State 1 to State 0, $Z_R(t)$, is well modelled through Weibull or Lognormal distribution, yet the Lognormal is a better fit when tested with the use of a negative log-likelihood. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled by Weibull or Lognormal distribution, yet the Weibull is a better fit when tested using the negative log-likelihood.

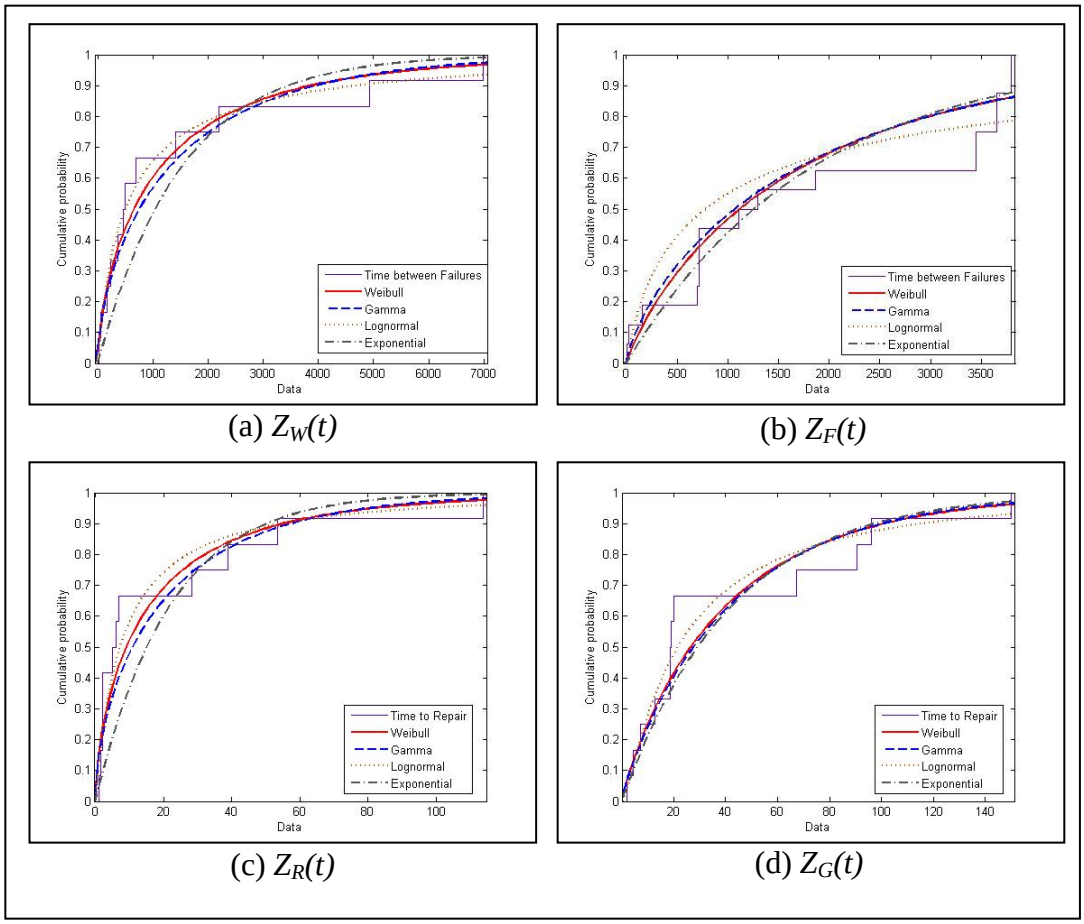


Figure 39: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource A Site 1.

Resource B Site One:

As can be seen from Figure 40, visual inspection shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Gamma distribution, yet the Weibull is a better fit when tested using negative log-likelihood. The time between transitions from State 0 to 2 is, $Z_F(t)$, well modelled by Weibull or Gamma, yet the Weibull is a better fit when tested using the negative log-likelihood. Moreover, the time between transitions from State 1 to State 0, $Z_R(t)$, is well modelled by Lognormal distribution; both the visual inspection and the negative log-likelihood test confirm this. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled by Weibull or Lognormal distribution, although the Weibull is a better fit when tested using negative log-likelihood.

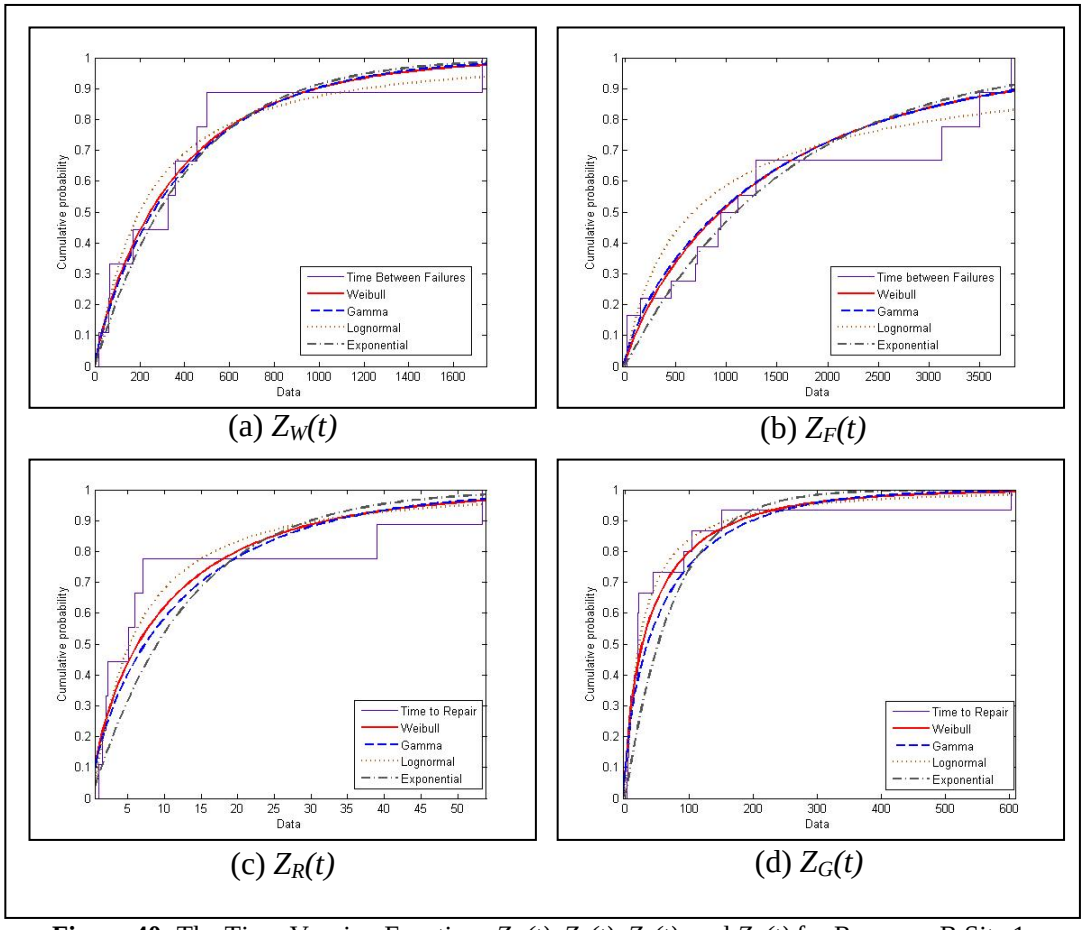


Figure 40: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource B Site 1.

Resource C Site One:

From Figure 41, visual inspection shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Lognormal distribution, although the Weibull is a better fit when tested using negative log-likelihood. The time between transitions from State 0 to 2, $Z_F(t)$, is well modelled by Weibull or Gamma, yet the Weibull is a better fit when tested using the negative log-likelihood. Furthermore, the time between transitions from State 1 to State 0, $Z_R(t)$, is well modelled Lognormal or Weibull distribution, yet the Weibull is a better fit when tested using the negative log-likelihood. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled through Weibull or Lognormal distribution, yet the Lognormal is a better fit when tested using negative log-likelihood.

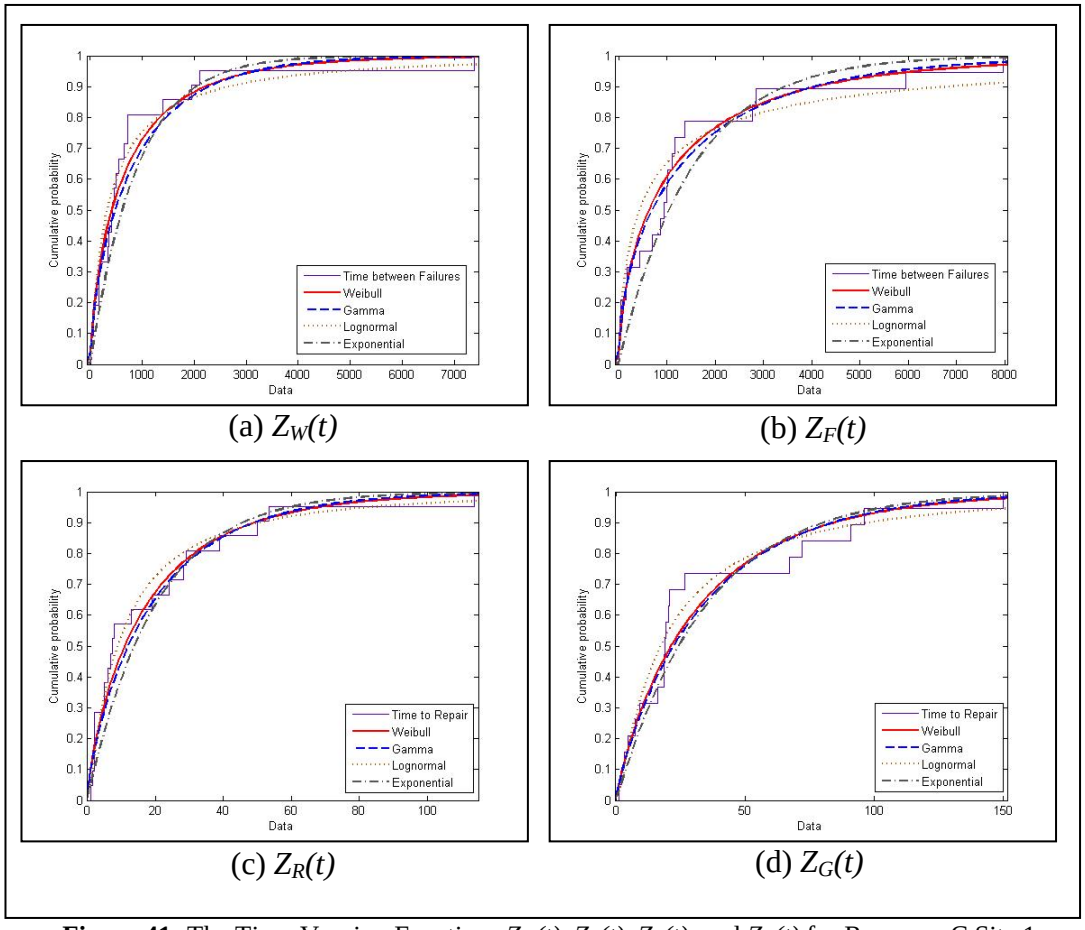
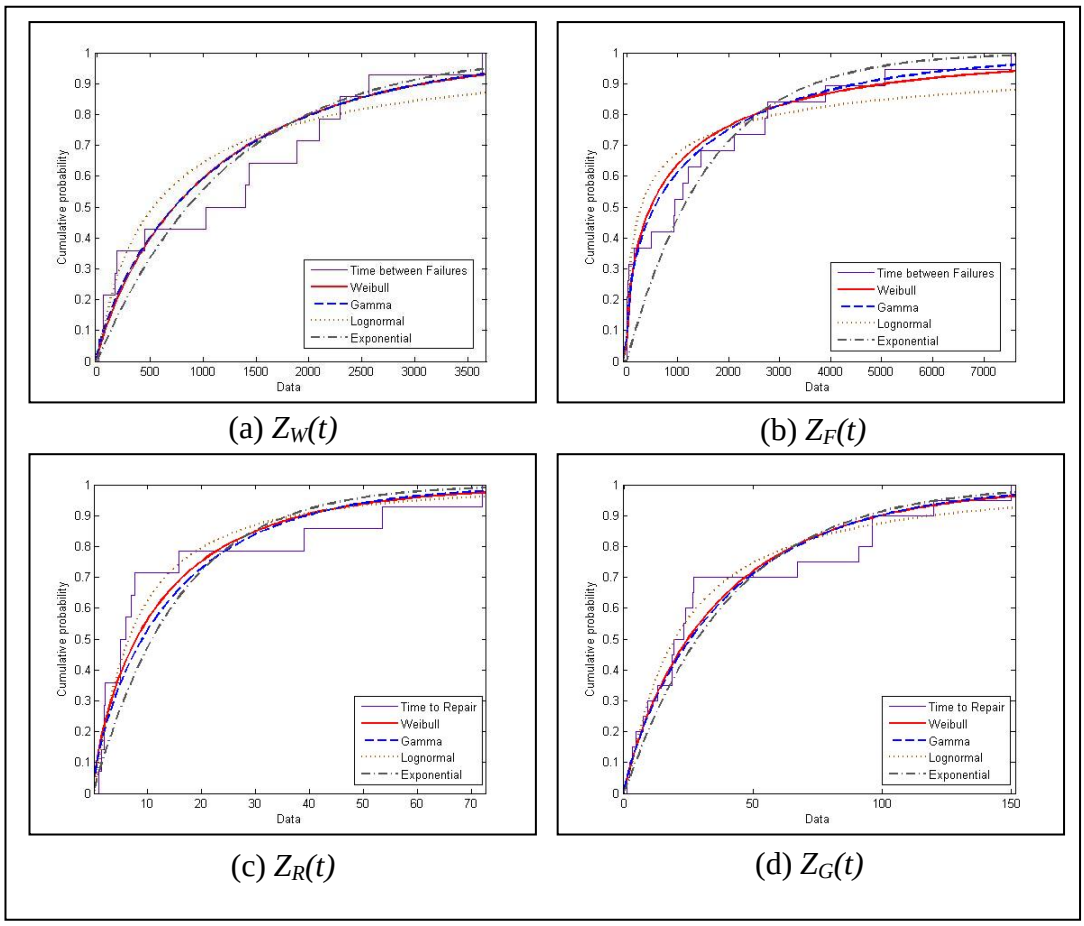


Figure 41: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource C Site 1.

Resource D Site One:

Figure 42 shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Gamma; both distributions create an equally good visual fit and the same negative log-likelihood. The time between transitions from State 0 to 2, $Z_{F(}t)$, is also well modelled by Weibull or Gamma, although the Gamma is a better fit when tested using the negative log-likelihood. The time between transitions from State 1 to State 0, $Z_R(t)$, is well modelled Lognormal or Weibull distribution, yet the Lognormal is a better fit when tested using the negative log-likelihood. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled by Weibull or Lognormal distribution, though the Weibull is a better fit when tested using negative log-likelihood.



(a) $Z_W(t)$ (b) $Z_F(t)$

(c) $Z_R(t)$ (d) $Z_G(t)$

Figure 42: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource D Site 1.

Resource A Site Two:

Figure 43 shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Exponential, although the Weibull is a better fit when tested using negative log-likelihood. The time between transitions from State 0 to 2, $Z_F(t)$, is also well modelled by Weibull or Gamma; both distributions create an equally good visual fit and the same negative log-likelihood. The time between transitions from State 1 to State 0, $Z_R(t)$, is well modelled Gamma or Weibull distribution, although the Weibull is a better fit when tested using the negative log-likelihood. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled by Weibull or Lognormal distribution, yet the Lognormal is a better fit when tested using negative log-likelihood.

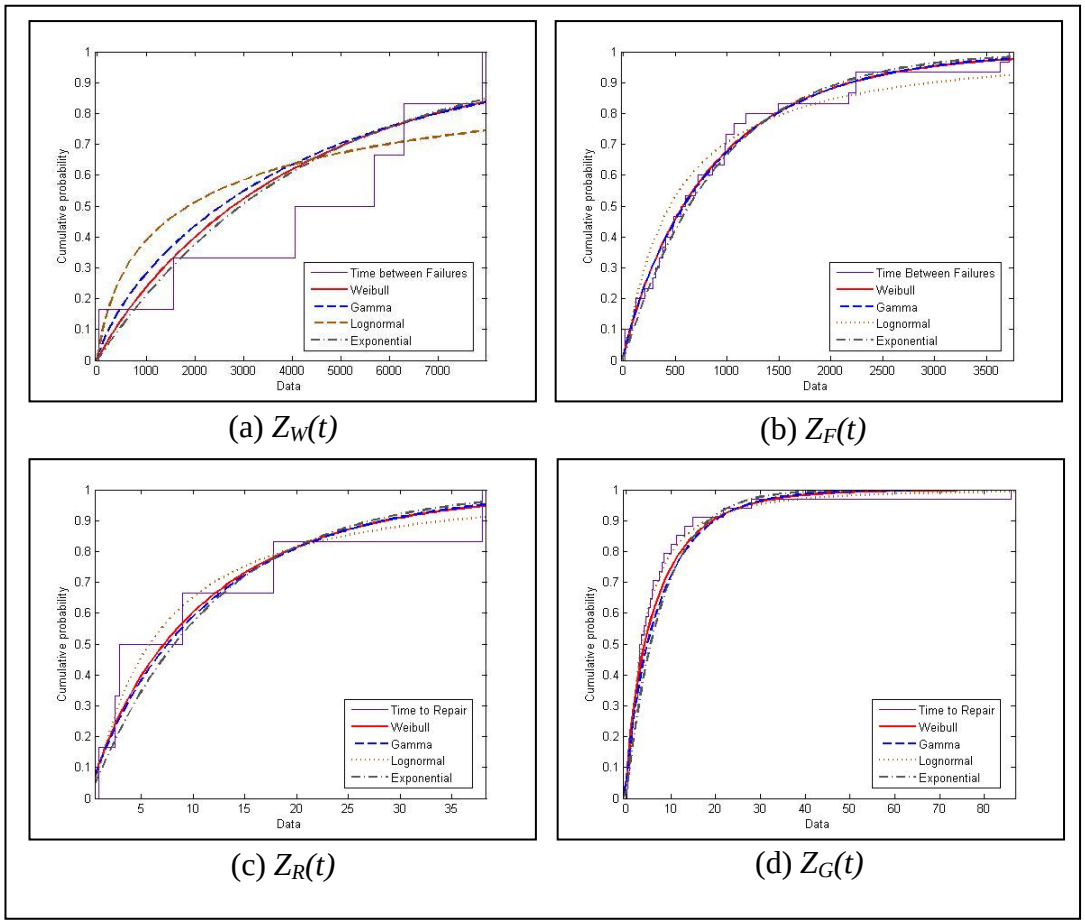
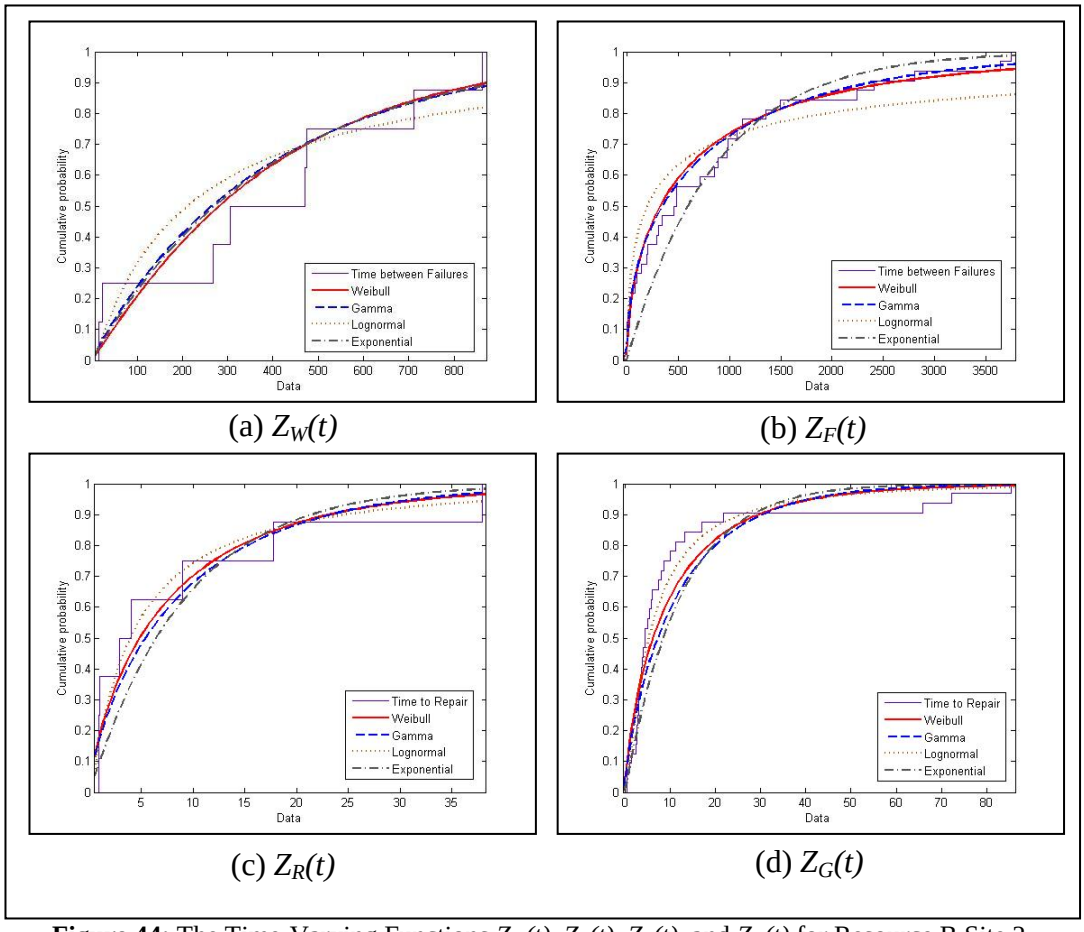


Figure 43: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource A Site 2.

Resource B Site Two:

Figure 44 illustrates that the time between transitions from State 0 to 1 is well modelled by Weibull or Exponential, although the Weibull is considered to be a better fit when tested using negative log-likelihood. The time between transitions from State 0 to 2 is well modelled by Weibull or Gamma, although the Gamma is a better fit when tested using the negative log-likelihood. The time between transitions from State 1 to State 0 is well modelled Lognormal or Weibull distribution, but the Weibull is a better fit when tested using the negative log-likelihood. The time between transitions from State 2 to State 0 is well modelled by Weibull or Lognormal distribution, yet the Lognormal is a better fit when tested using negative log-likelihood.



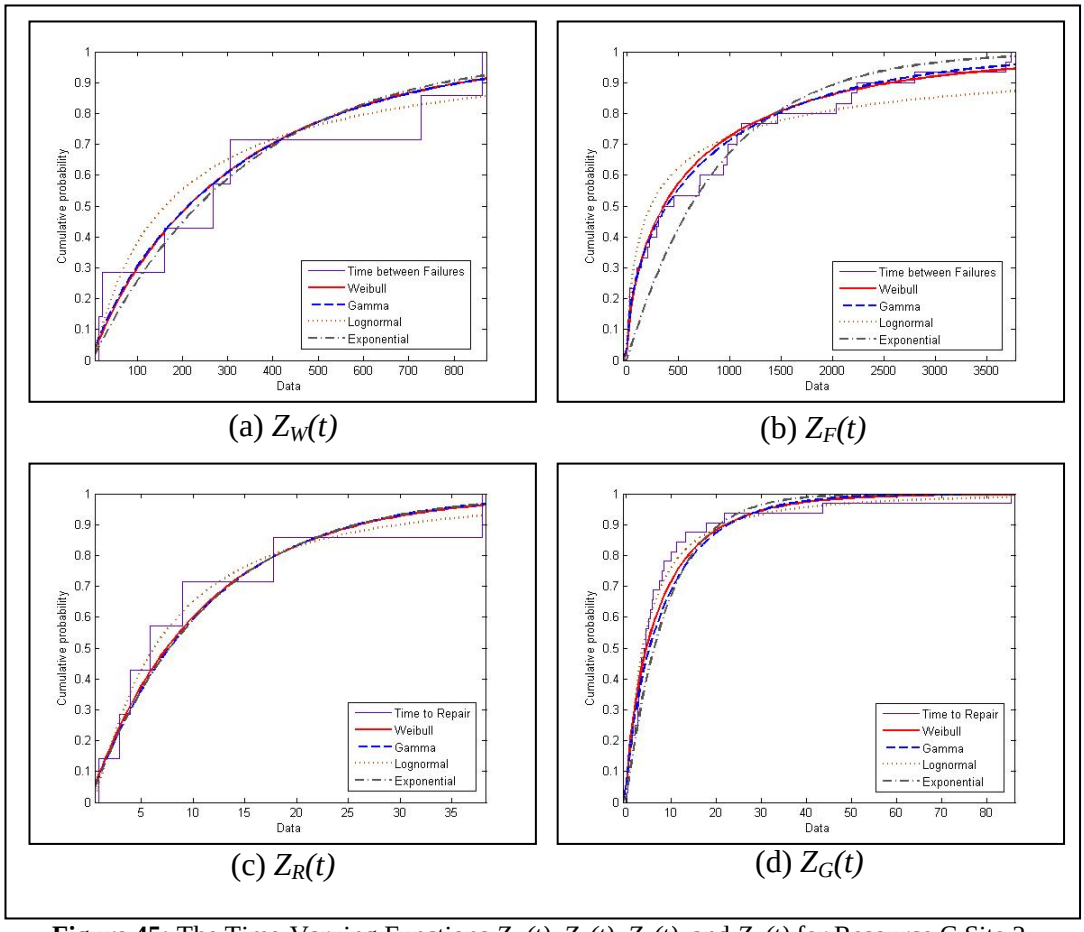
(a) $Z_W(t)$ (b) $Z_F(t)$

(c) $Z_R(t)$ (d) $Z_G(t)$

Figure 44: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource B Site 2.

Resource C Site Two:

Figure 45 shows that the time between transitions from State 0 to 1, $Z_W(t)$, is well modelled by Weibull or Gamma; both distributions create an equally good visual fit and the same negative log-likelihood. The time between transitions from State 0 to 2, $Z_F(t)$, is well modelled by Weibull or Gamma, although the Gamma is a better fit when tested using the negative log-likelihood. The time between transitions from State 1 to State 0, $Z_R(t)$, is well modelled Lognormal or Weibull distribution, but the Weibull is a better fit when tested using the negative log-likelihood. The time between transitions from State 2 to State 0, $Z_G(t)$, is well modelled by Weibull or Lognormal distribution, although the Lognormal is a better fit when tested using negative log-likelihood.



(a) $Z_W(t)$ (b) $Z_F(t)$

(c) $Z_R(t)$ (d) $Z_G(t)$

Figure 45: The Time-Varying Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ for Resource C Site 2.

As can be seen from the above results, the function $Z_W(t)$ is modelled by a Weibull distribution since the transition from State 0 to State 1 in all seven resources is best fitted by the Weibull distribution. The function $Z_F(t)$ is similarly modelled by a Weibull distribution, owing to the fact that the transition from State 0 to State 2 in four out of the seven resources achieves best fit through a Weibull distribution, whilst for the remaining three the Gamma distribution fit the data slightly better than the Weibull distribution, but still provides a good fit. The function $Z_R(t)$ is also modelled by a Weibull distribution since the transition from State 1 to State 0 in four out of the seven resources achieves a best fit from a Weibull distribution, whilst for the other three, the Lognormal distribution fit the data slightly better than the Weibull distribution. Finally, the function $Z_G(t)$ is also modelled by a Weibull distribution, although the transition from State 2 to State 0 in three out of the seven resources achieves best fit through Weibull distribution. The reason for this is that, in the case of the other four resources, the Lognormal distribution was only a slightly better fit than the Weibull distribution.

Table 11 shows the individual resources along with the best distribution fit for the four transition functions.

Table 11: The Best Fit Distribution for the Transition Functions.

		$Z_W(t)$	$Z_F(t)$	$Z_R(t)$	$Z_G(t)$
Site One	*Resource A*	Weibull	Weibull	Lognormal	Weibull
	Resource B	Weibull	Weibull	Lognormal	Weibull
	Resource C	Weibull	Weibull	Weibull	Lognormal
	Resource D	Weibull	Gamma	Lognormal	Weibull
Site Two	*Resource A*	Weibull	Weibull	Weibull	Lognormal
	Resource B	Weibull	Gamma	Weibull	Lognormal
	Resource C	Weibull	Gamma	Weibull	Lognormal

5.3 Developing the Risk Assessment Model

In the previous section, the time between events was found to follow a Weibull distribution; therefore, the time-varying functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ are

based on a Weibull probability density function (pdf), with unique shape α and scale λ values for each function.

$$Z_W(t) = \alpha_W \lambda_W \ (\lambda_W \ t)^{(\alpha_W - 1)} \ e^{-(\lambda_W \ t)^{\alpha_W}}$$

$$Z_R(t) = \alpha_R \ \lambda_R \ (\lambda_R \ t)^{(\alpha_R - 1)} \ e^{-(\lambda_R \ t)^{\alpha_R}}$$

$$Z_F(t) = \alpha_F \ \lambda_F \ (\lambda_F \ t)^{(\alpha_F - 1)} \ e^{-(\lambda_F \ t)^{\alpha_F}}$$

$$Z_G(t) = \alpha_G \ \lambda_G \ (\lambda_G \ t)^{(\alpha_G - 1)} \ e^{-(\lambda_G \ t)^{\alpha_G}}$$

Continuous time-varying Markov models are difficult and complex to solve. The numerical integration technique is one method for solving the model, whilst an alternative method involves approximating the continuous-time process with discrete-time equivalents [205]. The latter will be used as numerical integration involves some degree of approximation.

Figure 46 shows the resulting discrete-time Markov model for time step Δt. Since more than one transition may occur during a time step, the model must take into account the joint probability of state transition.

The state transition probabilities for the discrete-time Markov model changes over time; therefore, we need to drive an expression for *A(n)*, *B(n)*, *C(n)*, *D(n)*, and *E(n)*. The model we drive is based on models developed by Howard [204] and Siewiorek and Swarz [205].

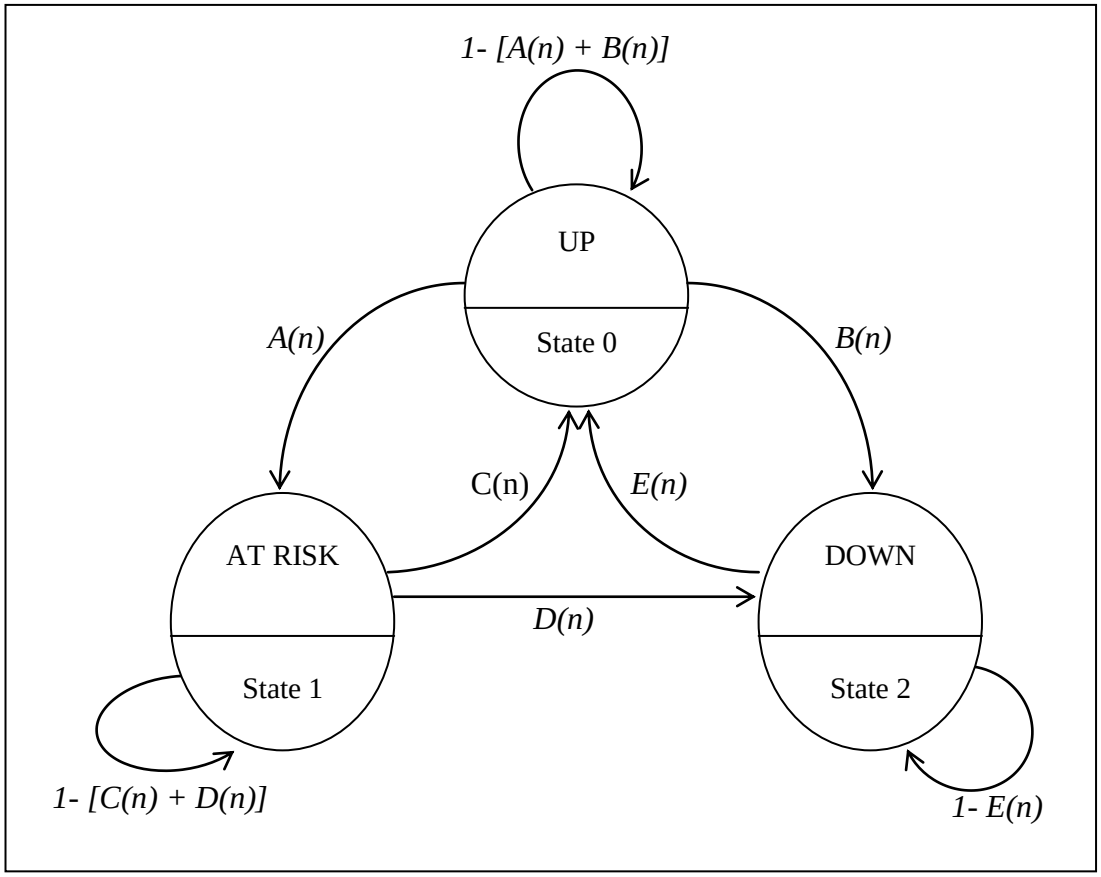


Figure 46: Discrete-time Markov Model for Resource Availability.

The interest is in calculating the probability transition equations, in which q_{ij} *(m, n)* is the probability that the system is in state *j* at time *n* given that it was in state *i* at time *m* ($m \leq n$). With this notation, in matrix form the Chapman-Kolmogorov equation [204] is:

$$Q(m, n) = Q(m, k)\, Q(k, n) \quad m \leq k \leq n$$

Letting $k = n - 1$,

$$Q(m, n) = Q(m, n - 1)\, Q(n - 1, n)$$

Defining $P(n) = Q(n, n + 1)$,

$$Q(m, n) = Q(m, n - 1)P(n - 1)$$

Expanding the equation recursively

$$Q(m, n) = Q(m, n - 2)\, P(n - 2)\, P(n - 1)$$
$$= Q(m, n - 3)\, P(n - 3)\, P(n - 2)\, P(n - 1)$$

Yielding the final solution

$$Q(m, n) = \prod_{i=m}^{n-1} P(i)$$

In order to convert from continuous-time probability functions to discrete-time probability function, a discrete-time probability distribution must be established that corresponds to the continuous-time distribution. The corresponding parameters can then be calculated for the desired time-step Δt. Furthermore, a discrete-time approximation has to consider the probability of two failures during the same interval; the time-varying reliability functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$ are based on a Weibull probability density function.

$$pdf = f(t) = \alpha\lambda(\lambda t)^{\alpha-1} e^{-(\lambda t)^{\alpha}}$$

The corresponding discrete Weibull function, probability mass function, is:

$$pmf = f(k) = q^{k^{\alpha}} - q^{(k+1)^{\alpha}}$$

Given that $f(k)$ is defined as the probability of an event occurring between time Δt and time $(k + 1)\ \Delta t$ for some chosen interval size Δt, the probability mass function can be expressed as:

$$f(k) = P[\text{no event by } k\Delta t] - P[\text{no event by } (k + 1)\Delta t]$$

$$f(k) = R(k) - R(k+1)$$

$R(k)$ is the reliability function. By substituting the continuous-time equivalents yields:

$$f(k) = R(k\Delta t) - R[(k + 1)\ \Delta t]$$

$$f(k) = e^{-(\lambda k\Delta t)^{\alpha}} - e^{-[\lambda(k+1)\Delta t]^{\alpha}}$$

By rearranging terms, we can find that:

$$q = e^{-(\lambda\Delta t)^{\alpha}}$$

The probability mass functions $Z_W(n)$, $Z_R(n)$, $Z_F(n)$, and $Z_G(n)$ provides the reliability for a discrete time step $n = t_n/\Delta t$. The time-varying functions are:

$$q_W = e^{-(\lambda_W \Delta t)^{\alpha_W}}$$

$$Z_W(n) = 1 - q_W{}^{(n+1)^{\alpha_W} - n^{\alpha_W}}$$

$$q_R = e^{-(\lambda_R \Delta t)^{\alpha_R}}$$

$$Z_R(n) = 1 - q_R{}^{(n+1)^{\alpha_R} - n^{\alpha_R}}$$

$$q_F = e^{-(\lambda_F \Delta t)^{\alpha_F}}$$

$$Z_F(n) = 1 - q_F^{(n+1)^{\alpha_F} - n^{\alpha_F}}$$

$$q_G = e^{-(\lambda_G \Delta t)^{\alpha_G}}$$

$$Z_G(n) = 1 - q_G^{(n+1)^{\alpha_G} - n^{\alpha_G}}$$

The transition probability functions in Figure 46, which represent the probability of transition from one state to another state, are:

$$A(n) = [1 - Z_F(n)]\ Z_W(n)$$

$$B(n) = [1 - Z_W(n)]\ Z_F(n)$$

$$C(n) = [1 - Z_F(n)]\ Z_R(n)$$

$$D(n) = [1 - Z_R(n)]\ Z_F(n)$$

$$E(n) = Z_G(n)$$

The transition probability matrix

$$\boldsymbol{P}(n) = \begin{bmatrix} 1 - [A(n) + B(n)] & A(n) & B(n) \\ C(n) & 1 - [C(n) + D(n)] & D(n) \\ E(n) & 0 & 1 - E(n) \end{bmatrix}$$

- *A(n)* is the probability of not transiting to *Down* and the probability of transiting from *Up* to *At Risk*,
- *B(n)* is the probability of not transiting to *At Risk* and the probability of transiting from *Up* to *Down*,
- *C(n)* is the probability of not failing and transiting from *At Risk* to *Up*,
- *D(n)* is the probability of not been recovered and transiting *Down*,
- *E(n)* is the probability of repairing the system and transiting from *Down* to *Up*.

Taking into account that $\mathbf{P}_{i,j}$ is the probability of a transition from state *i* to state *j*, it can then be stated that the probability of transition $\mathbf{P}_{0,0}$ is the probability of remaining in State 0, which is 1 – the probability of leaving State 0, hence 1 – [*A(n) + B(n)*]. The same can then be applied for the probability of transition $\mathbf{P}_{1,1}$ and $\mathbf{P}_{2,2}$.

P(*n*) can be used to compute instantaneous or the point risk of failure, which is the probability that the system will not be operational at any random time *t*. However, the most important is the duration risk of failure, which is the probability

that the system will not be operational for the entire duration (e.g. job execution time). Computing duration risk of failure is an iterative process. Accordingly, applying the appropriate values for α and λ, starting at T = start time, **P**(*n*) is computed forward for successive values of *n* until the desired finish time $t = n\,\Delta t$ is reached.

5.4 Experimental Results and Validation

Adopting the technique described in the previous section, the transition matrix **P**(n) is computed for each resource using the data from GOCDB with Δt = 1 hour. Since Grid jobs usually require long execution times, Δt should be selected accordingly. However, very long Δt lowers the accuracy of the model, since a state transition is not promptly recorded. On the other hand, short Δt has the overhead of calculating **P**(*n*) multiple times, despite the probability of transition not changing. Therefore, Δt was selected to be 1 hour.

The observed risk of failure was calculated using the data from the last 6 months of 2010. There are two reasons for selecting 6 months as the time-span:

1. The resources failure data used to calculate the model span for three years; and
2. The Weibull shape parameter for resource failures is less than 1, which means that, following a failure, the risk of seeing one soon increases; therefore, a short time-span does not reflect the true behaviour of the resource failures.

Table 12 shows, for the resources considered, the values of the Weibull shape α and scale λ parameters for the reliability functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$. The MLE was used to estimate the parameters. The risk of failure is calculated as the sum of the probability of transitioning from *Up* to *At Risk* and the probability of transitioning from *Up* to *Down*.

The data from GOCDB is used to validate the predicted risk of failure. The observed risk of failure is defined as:

$$Observed\ ROF = \frac{The\ time\ the\ resource\ is\ down}{The\ time\ the\ resource\ is\ down + the\ time\ it\ is\ up}$$

The observed risk of failure was calculated using the data from the last 6 months of 2010. There are two reasons for selecting 6 months as the time-span:

3. The resources failure data used to calculate the model span for three years; and
4. The Weibull shape parameter for resource failures is less than 1, which means that, following a failure, the risk of seeing one soon increases; therefore, a short time-span does not reflect the true behaviour of the resource failures.

Table 12: The Shape α and Scale λ Parameters for the Functions $Z_W(t)$, $Z_R(t)$, $Z_F(t)$, and $Z_G(t)$.

		$Z_W(t)$		$Z_F(t)$		$Z_R(t)$		$Z_G(t)$	
		α	λ	α	λ	A	λ	A	λ
Site One	*A*	0.6741	1124.29	0.6002	1818	0.665	15.784	0.899	40.08
	B	0.8616	376.63	0.6409	1385.26	0.7385	10.454	0.5779	47.05
	C	0.7154	691.27	0.6384	1113.28	0.8022	17.387	0.8708	32.37
	D	0.8326	1138.13	0.6236	974.053	0.7565	12.936	0.8610	37.80
Site Two	*A*	0.5930	4160.27	0.8959	866.254	0.8715	11.014	0.7814	6.676
	B	1.0563	398.589	0.6806	613.096	0.7679	7.8319	0.6767	9.946
	C	0.8937	321.602	0.6930	657.811	0.9098	10.984	0.7593	7.392

Figures 47, 48, 49, & 50 show the predicted one-day duration risk of failure over a number of days, as well as the observed risk of failure for resources A, B, C and D from Site 1 correspondingly. Visual inspection indicates that the observed and predicted risks of failure are comparable. Figures 51, 52 & 53 show the predicted one-day duration risk of failure over a number of days, as well as the observed risk of failure for resources A, B and C from Site 2 correspondingly. Visual inspection indicates that the observed and predicted risks of failure are also comparable.

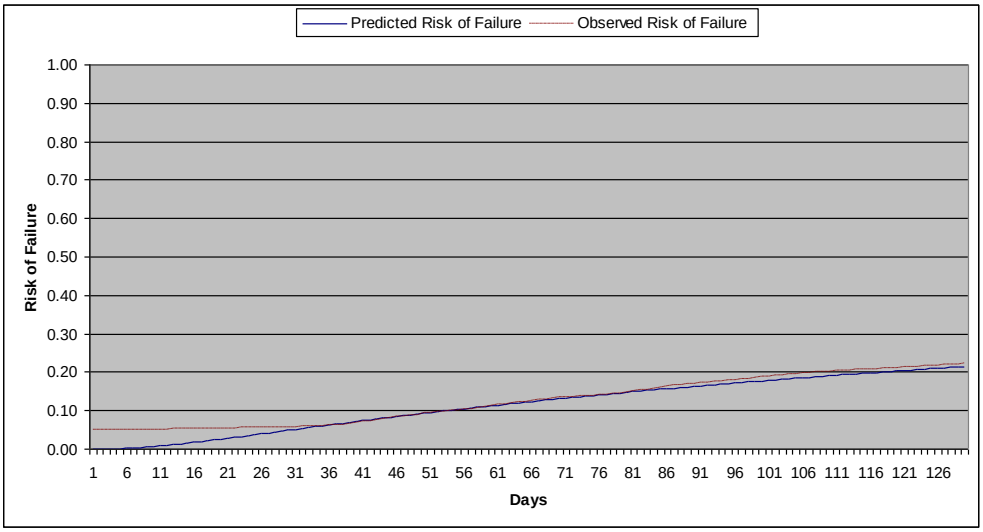


Figure 47: Predicted & Observed Risk of Failure for Resource A Site 1.

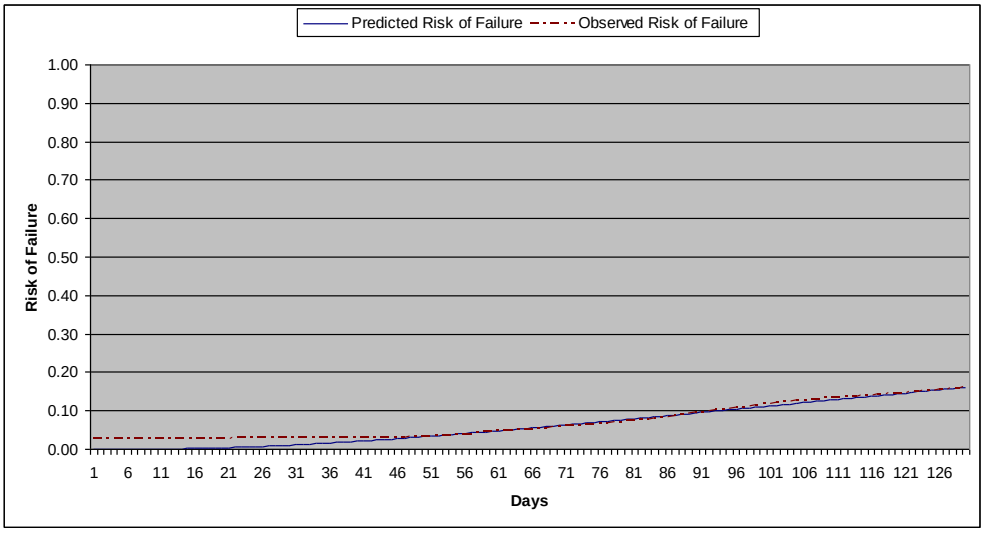


Figure 48: Predicted & Observed Risk of Failure for Resource B Site 1.

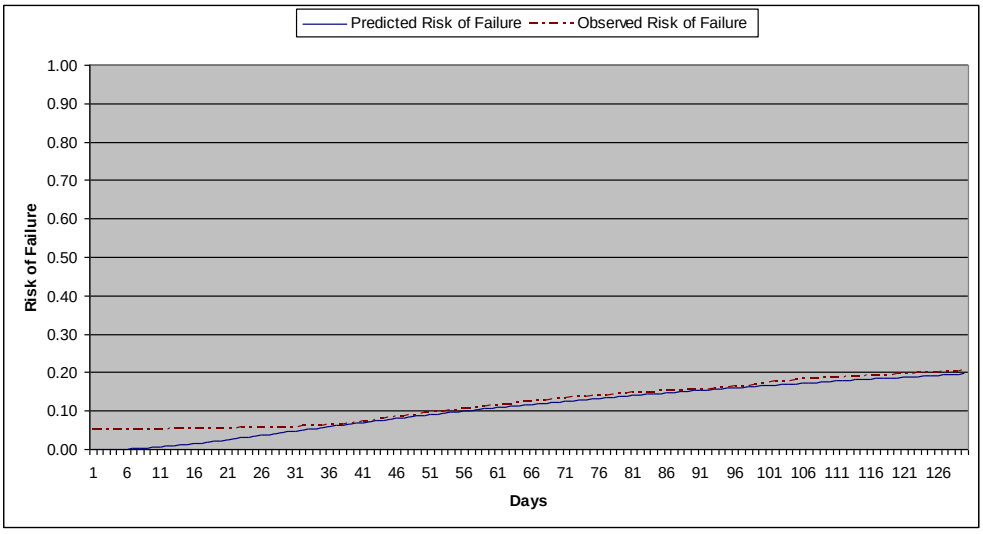


Figure 49: Predicted & Observed Risk of Failure for Resource C Site 1.

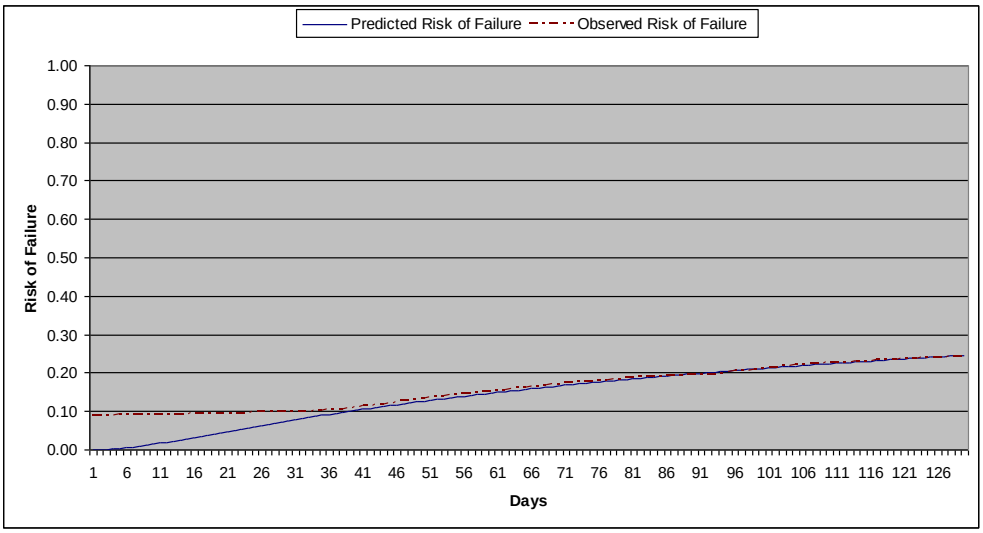


Figure 50: Predicted & Observed Risk of Failure for Resource D Site 1.

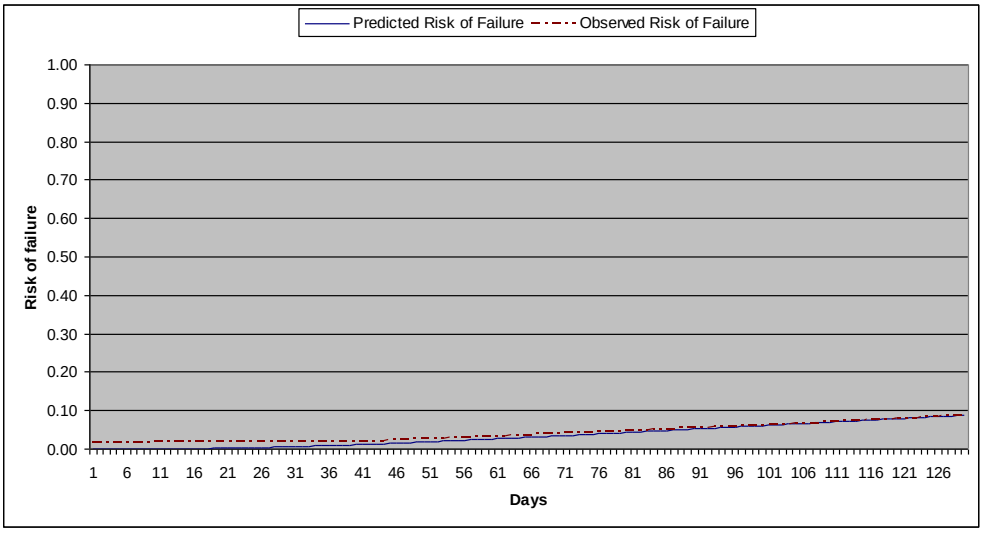


Figure 51: Predicted & Observed Risk of Failure for Resource A Site 2.

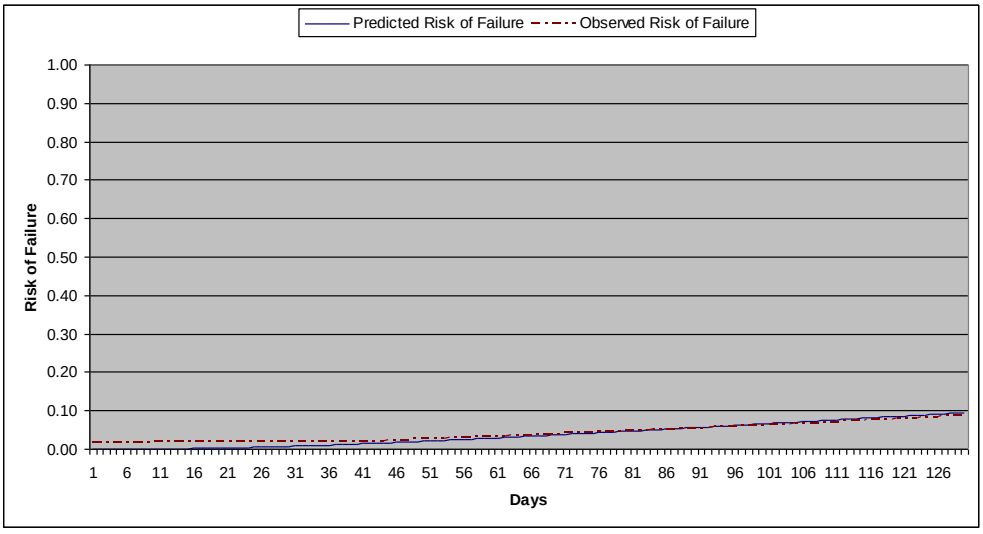


Figure 52: Predicted & Observed Risk of Failure for Resource B Site 2.

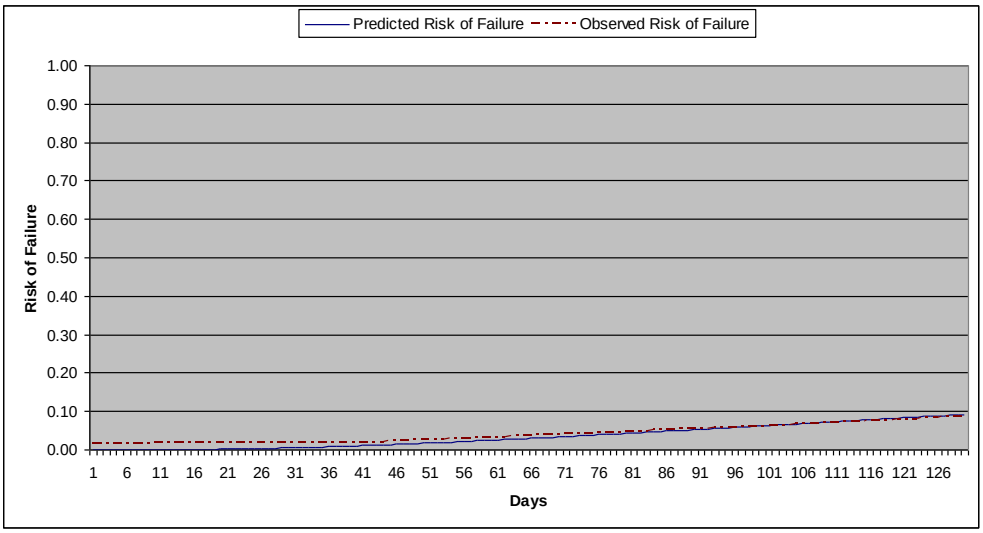


Figure 53: Predicted & Observed Risk of Failure for Resource C Site 2.

In order to validate that the predicted risk of failure is a true projection of the resource risk of failure (observed risk of failure), the two-sample t test—also known as Independent-Samples T Test—is used to compare the means of the two groups (observed and predicted risk of failure). The t test is used to compare exactly two groups, but differs to the Analysis of Variance (ANOVA) test, which compares three or more groups at one time [206].

The interest is to show that there is no difference between the predicted risk of failure and the observed risk of failure; however, it is impossible statistically to demonstrate that a statement is true. In actual fact, statistical techniques are much better at indicating that a statement is not true. Let the null hypothesis be there is no difference between the predicted and observed risk of failure. The alternative hypothesis is that there is a difference between the two.

The t test shows that the difference between the predicted and observed risk of failure is considered to be not statistically significant with P= 0.1636, P= 0.3491, P= 0.0935, and P= 0.0564, for site one resources, and P= 0.0556, P= 0.3827 and P= 0.0909 for site two resources (see Appendix B for the t test tables). Therefore, the null hypothesis is not rejected and, by default, the alternative hypothesis that there is a difference between the predicted and observed risk of failure is not supported. Thus, the conclusion is that there is no difference between the predicted and observed risk of failure.

From the above figures and the results of the t test, the conclusion is that the risk assessment model predicts accurately the resources risk of failure. Therefore,

the Grid resource provider can integrate the risk assessment model in order to compute the risk of resources failure.

5.5 Ranking Grid Resources and Planning Future Investments

The Grid resources ROF is unavoidable and, as such, ranking the resources with respect to their ROF is one of the most important outcomes of the risk assessment process. Ranking is simply arranging the resources based on their increasing or decreasing ROF. For Grid resources, the ranking is based on increasing ROF since resources with low ROF are better than resources with higher ROF. With this in mind, Figure 54 shows the predicted ROF of the seven resources over a duration of months. The ROF was computed, assuming all resources became available at exactly the same time $t = 0$. The X Axis represents the number of days, starting from Day 0, and the Y Axis represents the ROF. Moreover, Figure 55 illustrates the resources ROF from two randomly selected days—Day 30 and Day 90. On Day 30, Resource C, Site 2 has the lowest ROF, and therefore ranked first. On Day 90, Resource A, Site 2 has the lowest ROF, and thus ranked first. An important observation from Figures 54 & 55 is that Site 1 resources' ROF is almost always higher than Site 2 resources' ROF. The primary cause for this may be the time to repair a failed resource at each site. In the case of Site 1, for example, the time to repair resources A, B C and D, on average, takes approximately 32 hours, 27 hours, 28 hours and 31 hours respectively. In the case of Site 2, the time to repair resources A, B and C, on average, takes approximately 7 hours, 15 hours and 9 hours respectively (see 4.3.4 Repair Time Analysis).

Table 13 shows the complete list of ranked resources.

Table 13: The Complete List of Resources Ranked Based on Resource ROF, for Day 30 and Day 90.

Rank	*Day 30*	*Day 90*
1	Resource C, Site Two	Resource A, Site Two
2	Resource A, Site Two	Resource C, Site Two
3	Resource B, Site Two	Resource B, Site Two

4	Resource B, Site One	Resource B, Site One
5	Resource C, Site One	Resource C, Site One
6	Resource A, Site One	Resource A, Site One
7	Resource D, Site One	Resource D, Site One

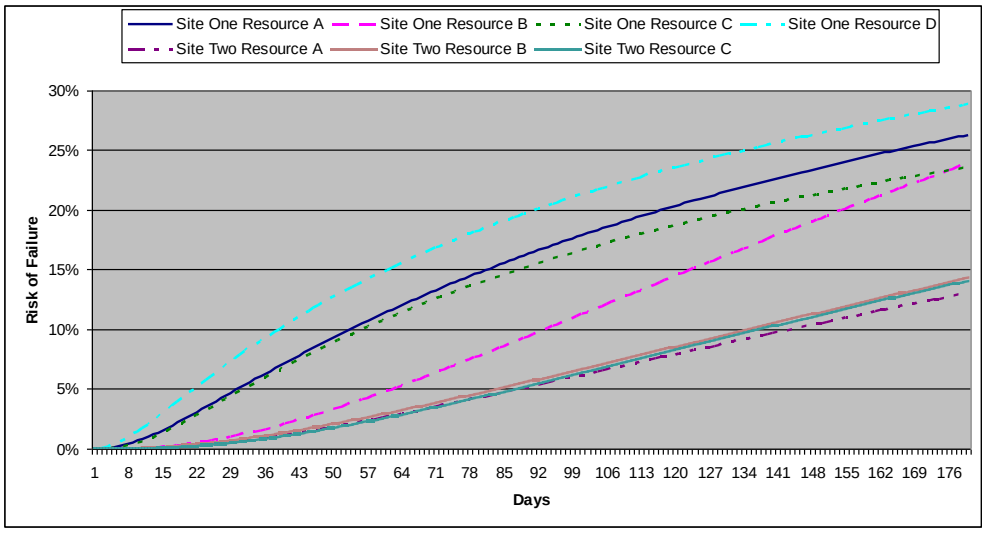


Figure 54: Resources Predicted ROF Over Days.

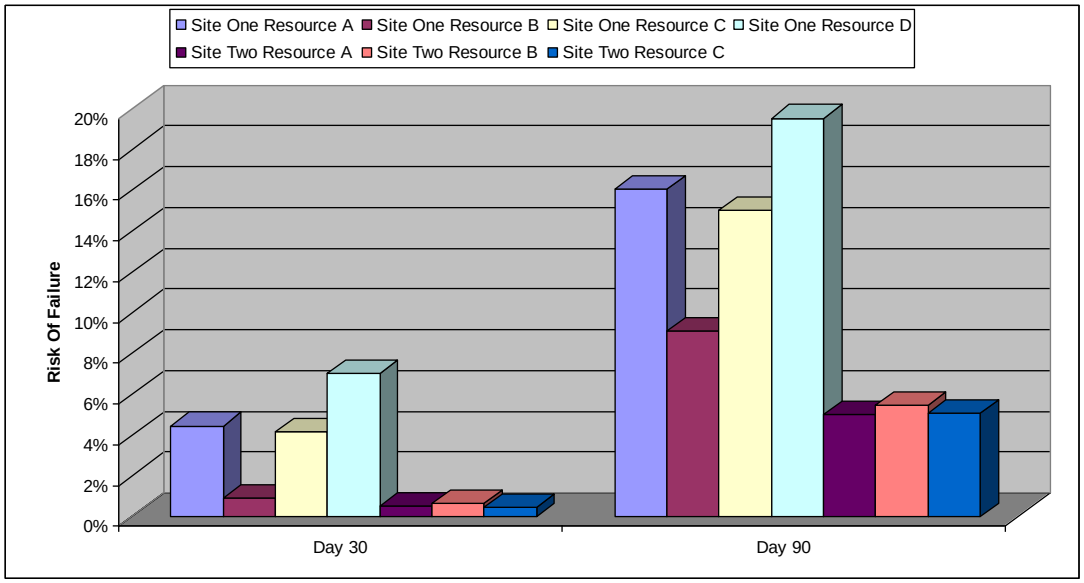


Figure 55: Resources Predicted ROF on Day 30 & Day 90.

In addition to ranking resources, the ROF model can be used to measure the significance of the effect of changes in the Grid resources and environment. The changes could be new or updated hardware, software, or even experience system administrators in order to lower resources' repair time. There are various techniques

for measuring this significance, the most commonly used of which is the 'one-at-a-time' method [9]. In this case, the assumptions and parameters are changed individually so as to measure the change in output.

The one-at-a-time methods, along with the ROF model, are very powerful tools for Grid providers to understand the limitations of current infrastructures and plan future investments. These tools are explained in the following example.

Assume a Grid resources provider would like to invest some money to improve the resources ROF. If the investment is on hardware, the provider then expects the time between hardware failures to increase by 50%. Similarly, the time between software failures is expected to increase by 50% if the investment is on software; if it was on experienced system administrators, the resources repair time would then decrease by 50%. The question is, which investment is the best? In other words, which results in lowering the resources ROF to the greatest extent. (The hardware and software failures were selected as they are the largest contributors to failures. See 4.3.3 Root Cause Breakdown.)

The procedure to answer this question for each resource is as follows:

1. Compute the ROF for the resource using the technique introduced in Section 5.3.
2. Decrease the time to repair the resource by 50%, and then compute the ROF after the change.
3. Return the time to repair to its original value and increase the time between hardware failures by 50%, and compute the ROF.
4. Return time between hardware failures to its original value, and increase the time between software failures by 50%, and then compute the ROF.

Figures 56, 57, 58, 59, 60, 61 & 62 show the original ROF, the ROF if the repair time is decreased by 50%, the ROF if the time between hardware failures is increased by 50% and the ROF if the time between hardware failures is increased by 50% over a number of days for Site 1, resource A, B, C, D, and Site 2, resource A, B and C correspondingly. The X Axis represents the number of days, starting from Day 0, whilst the Y Axis represents the ROF. Day 0 is the time when the resource became available—either after a scheduled maintenance or unscheduled failure.

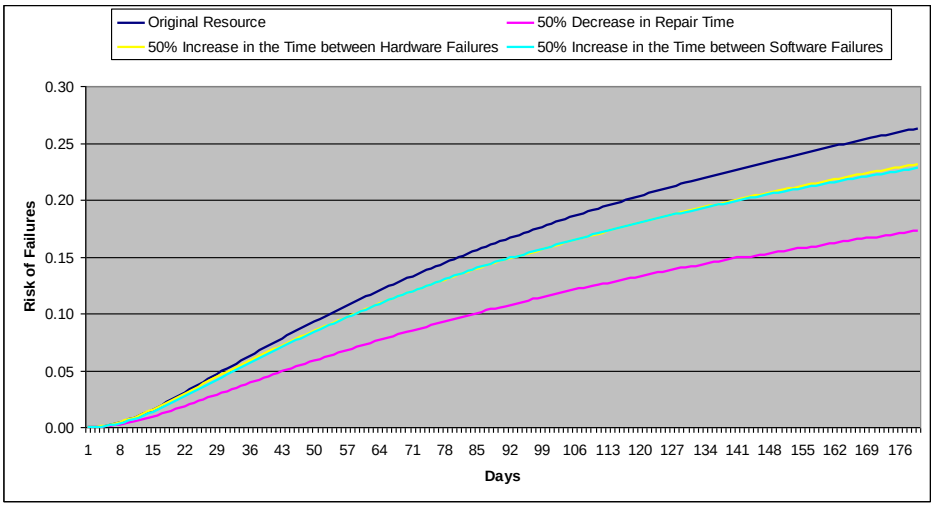


Figure 56: Investments effect on Resource A Site 1.

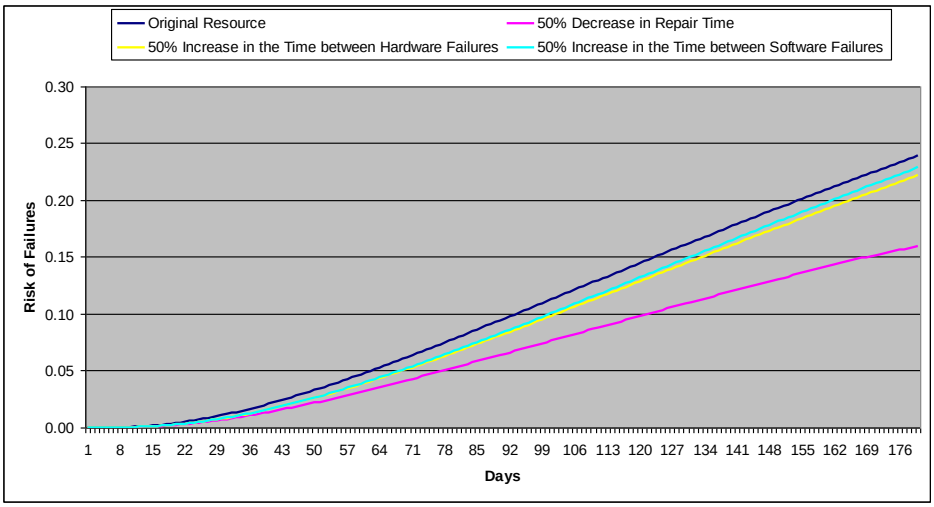


Figure 57: Investments effect on Resource B Site 1.

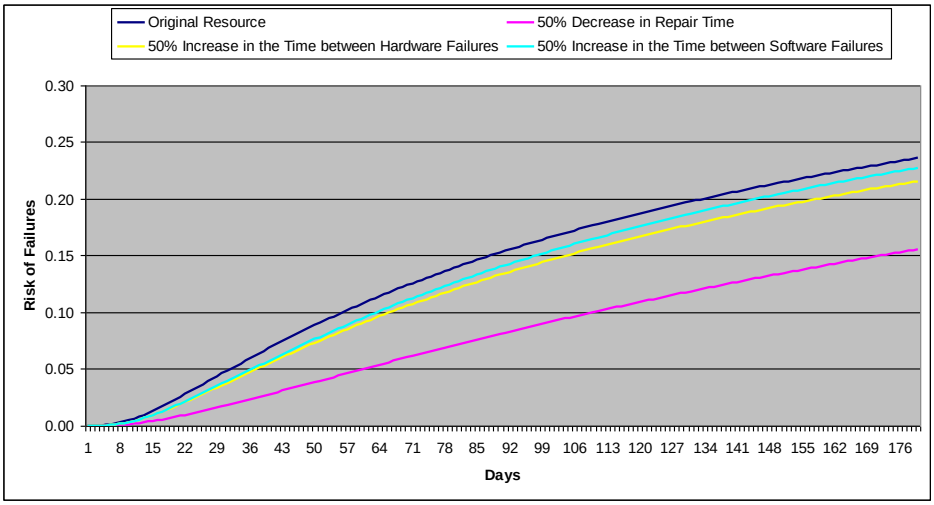


Figure 58: Investments effect on Resource C Site 1.

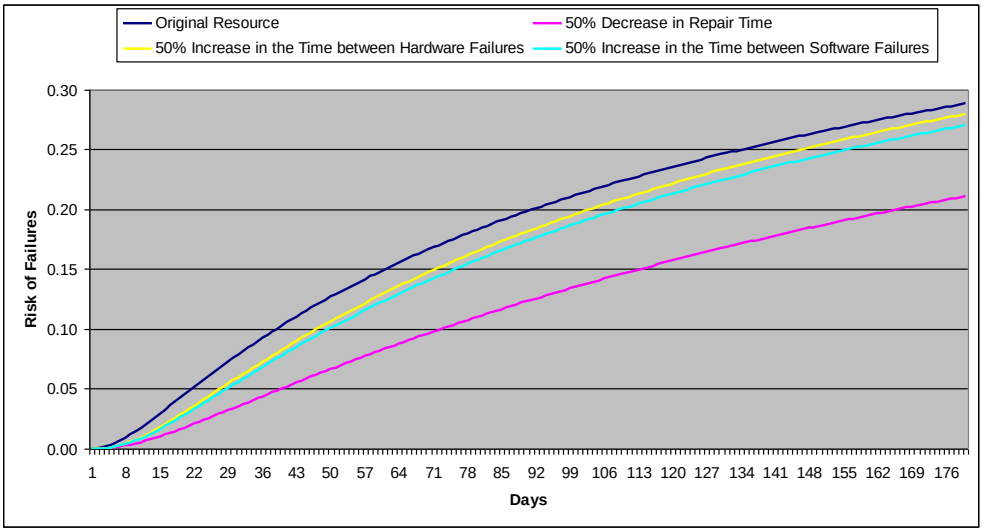


Figure 59: Investments effect on Resource D Site 1.

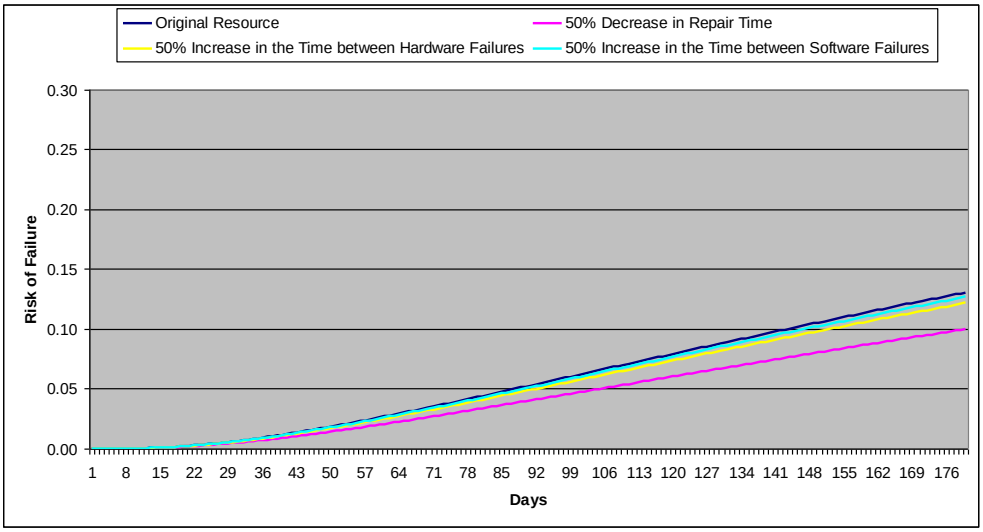


Figure 60: Investments effect on Resource A Site 2.

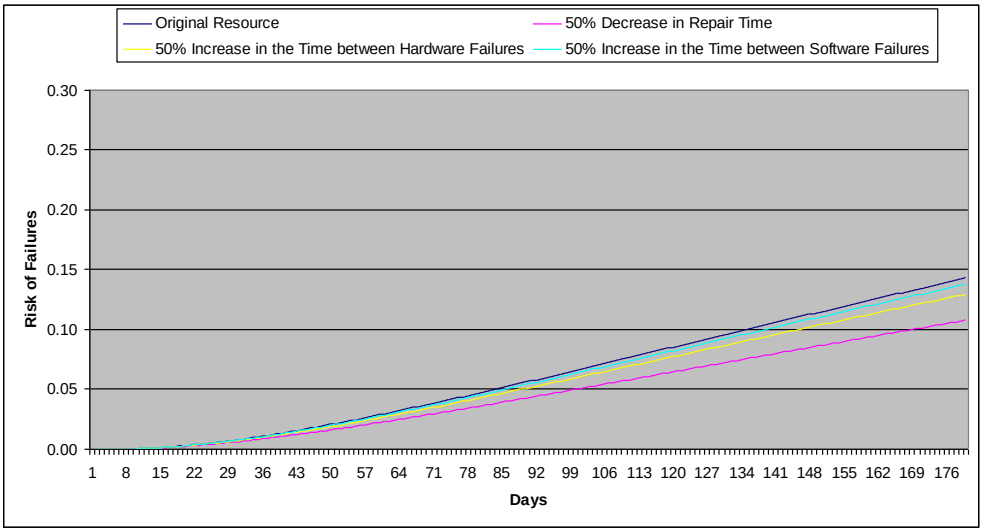


Figure 61: Investments effect on Resource B Site 2.

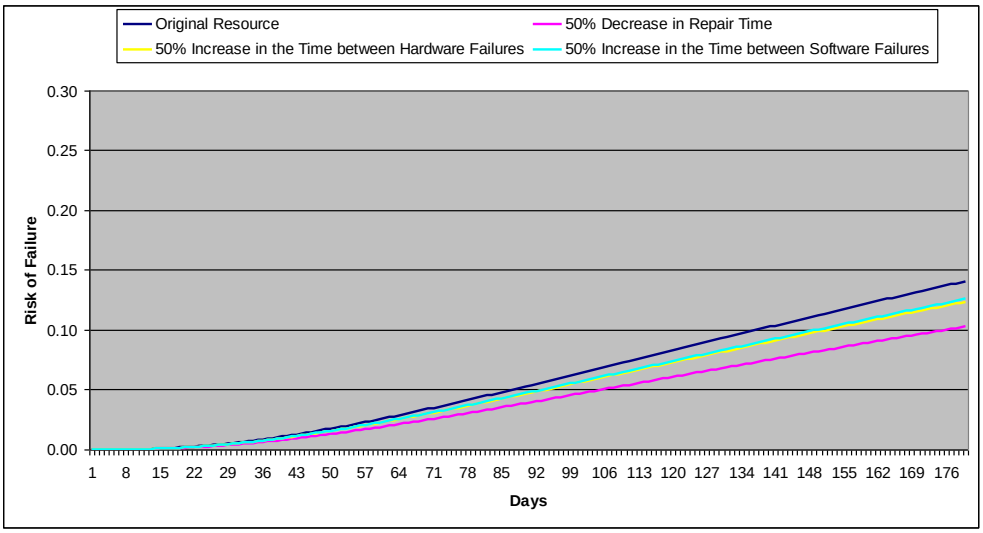


Figure 62: Investments effect on Resource C Site 2.

From the above figures, it can be observed that the investment in lowering the repair time is the most rewarding; this is because the repair time—in the case of all resources—is very high, even after the 50% decrease. Investment in hardware or software, at this stage, is not much rewarding as the benefit on lowering the ROF is limited.

5.6 Summary

In this chapter, a mathematical model for the prediction of the risk of failure, with the use of a discrete-time analytical model driven by distribution functions fitted to observed data, is presented.

The chapter begins by introducing availability models as a means for calculating the probability of failures or ROF. Two techniques for availability are discussed, namely Combinatorial Models and Markov Models. Grid resource availability is modelled by a three-state continuous time-varying Markov model. The state transition functions are driven from the distributions fitted to failure data. The transition functions were found to follow a Weibull distribution. The chapter then describes the method for solving the Markov model, which is to approximate the continuous-time process with discrete-time equivalents. The discrete time-varying Markov model is validated by comparing the predicted ROF with the observed ROF. Notably, both graphical and statistical evaluations are presented. The validation indicates that the difference between the observed ROF and the predicted ROF is not statistically significant. Finally, the chapter presents the use of the risk

assessment model to rank Grid resources and to measure the significance of the effect of changes in the Grid resources and environment.

Chapter 6

Using Resource ROF to Improve Scheduling

Grid environments provide computing platforms for solving large-scale computational and data-intensive problems in science, engineering, and commerce. They can be very cost-effective and easily scalable yet, owing to resource heterogeneity and to the lack of accurate resource information, scheduling jobs in such systems can be challenging. In this chapter, the problem of scheduling Bag of Tasks (BoT) application on Grid resources is modelled using Mixed Integer programming. An efficient algorithm for solving the scheduling problem is presented. The algorithm is evaluated with the use of a simulation, allowing a wide range of possible scenarios to be considered.

This chapter is organised as follows: Section 6.1 presents an overview of the scheduling problem and presents the Grid application model, as well as limitations of current scheduling algorithms. Section 6.2 provides the use of resources ROF to overcome the current algorithms limitations, and suggests a model to minimise the cost of executing a BoT job whilst guaranteeing the user's requirements. Section 6.3 presents the formal mathematical model and methods to compute the model optimal solution. A new scheduling algorithm, along with the algorithm pseudocode, is described in Section 6.4. Section 6.5 presents the evaluation of the algorithm through simulation. The simulation experiments' design, the resource model, the workload model, the experimental results, and sensitivity to the user constraints are presented and discussed in this section. Finally, Section 6.6 ends the chapter with a summary.

6.1 Overview

In the motivation scenario 4.1—as indeed in the real world—Grid users submit their application to resource providers through the use of SLAs. The SLA has the user application information, as well as the user requirements and constraints.

Notably, requirements can include the type of hardware, the type of operating system, or even a business objective, such as minimising the costs associated with executing the application. Moreover, a constraint could be the deadline by which the application results should be delivered. Once the resource provider receive an SLA, it is translated into an allocation problem whereby the application is allocated to resources for executing, ensuring that, during the execution time, the user requirements and constraints are being fulfilled.

The focus in this chapter is not directed on SLAs and their uses, but rather on the resource provider being able to schedule users' applications and accordingly guarantee their requirements and constraints. Therefore, in the remainder of this chapter, the assumption is that the resource provider's unit responsible for resource allocation—known as the scheduler—receives the user application, requirements and constraints in the required format for resources allocation.

6.1.1 Application Model and Scheduling

The type of applications which are executed on Grid systems can vary from long running computationally intensive simulations to high demand and high priority time critical transaction based executions, to real-time interactive visualisations (see 2.2 Grid Applications). Notably, the majority of these applications are sequential applications, often submitted in the form of Bags of Tasks (BoT). According to Iosup *et al*. [87] BoT jobs account for up to 96% of the CPU time consumed in Grid environments. BoT jobs are composed of sequential, independent tasks where there is no communication or dependencies amongst tasks. Examples of BoT jobs include Monte Carlo simulations, massive searches (such as key-breaking), image manipulation applications, data mining algorithms, and parameter-sweep applications [207]. Therefore, the type of applications which this thesis is targeting is BoT jobs.

Executing BoT jobs involves processing N independent tasks on M distributed resources where N is, typically, much larger than M. For each task $n \in N$ its computation time is known. Scheduling the tasks to resources appears simple, but complexities arise when users place their desired constraints. The job owners submit their BoT jobs and requirements in real time (in the reset of the chapter the job owners are referred to as users); therefore, the scheduler must find the tasks assignment efficiently and effectively for each user. The scheduling is carried out in

real-time, and the users' BoT are assigned as first in, first out (FIFO). If an assignment is found which has satisfied the user requirements, the user BoT job is then accepted; otherwise, the job is rejected.

Scheduling BoT jobs in Grid environments whilst guaranteeing the user constraints is an NP-hard problem [119]. A number of algorithms have been suggested for solving this problem (for more information about the algorithms see 2.6.2.3 Scheduling Algorithms). The available algorithms have a number of limitations, such as:

1. the algorithms only consider the deadline and cost constraints;
2. the algorithms assume the resource price is a function of performance. A more expensive resource is always faster, in term of processing speed, than a less expensive one. In the real world this assumption is invalid, because resources failures do occur;
3. the BoT jobs are assumed to be of the same level; accordingly, there is no distinction between critical and non-critical BoT jobs. What is meant by critical BoT jobs are the jobs that must be completed before a strict deadline; after this deadline, the results might be insignificant. Examples of critical BoT jobs include a researcher who needs the results of his/her BoT job before the submission deadline of a research conference or an organisation employee who needs the results of the BoT job before an important meeting. Presumably, users with critical BoT jobs are willing to pay more to ensure the jobs finish on time. Results of non-critical BoT jobs do not lose their significance after the deadline; therefore, the owners of such jobs would like to execute the jobs as inexpensively as possible rather than paying extra to ensure the deadlines are met; and finally,
4. the algorithms do not take into account the resource ROF. As a result, do not distinguish between resources with high ROF and low ROF.

6.2 Improving the Scheduling Algorithms

In Chapter 5, a mathematical model for estimating resources ROF was presented. The resource ROF is an important characteristic of the resource, and

should be considered when scheduling. Including the ROF in scheduling will address most of the limitations associated with the current BoT scheduling algorithms. The following points are the impacts of considering resources ROF on the limitations of current algorithms:

- current algorithms assume that resources are only identified by processing abilities and cost; the resources exhibit high availability and there are no resource failures. These are unrealistic assumptions as computer resources are prone to failures, with some failing more than others. Therefore, including resources ROF as part of the scheduling algorithms reflects the real world;
- current algorithms assume that the resource price per time unit is a function of processing ability. In the real world, however, ability is not the seldom function for pricing; this can be seen easily in the commercial world. For example, consider the ability to travel by air between two cities. If the price is a function of ability, then all flights should cost the same; unfortunately, however, they do not. Therefore, other factors for pricing Grid resources should be considered—one of which is the resource ROF. If two resources have the same processing ability but different ROF then, in theory, the resource with lower ROF is more expensive;
- current algorithms assume that there are no distinctions between the BoT jobs. Including the resources ROF enable the user to request the desired resources based on the job requirements. For example for a critical job the user request only resources with low ROF to ensure that the job requirements are fulfilled—even if these resources are more expensive. For non-critical jobs the user, might, request cheaper resources with higher ROF to minimize the cost of executing the jobs.

In this chapter, we present a new model for scheduling BoT jobs. The model objective is to minimise the cost of executing a BoT job. Two user constraints are considered to be the job deadline, and the resources ROF, i.e. which the user desires to use. The model considers the task's execution time on different resources, the resources prices, and the resources ROF. Moreover, the model takes into account that the resources are limited and some BoT jobs may be rejected.

To the best of our knowledge, none of the current scheduling algorithms address the issue of resources ROF. However we adopt some ideas from Buyya *et al.* [117] and Kumar *et al.* [119] algorithms to design the proposed algorithm.

6.3 Model Description

Scheduling BoT jobs to minimize the cost of execution while guaranteeing the user's requirements represent an optimization problem. Optimization refers to choosing the best elements from some set of available alternatives. Mathematical programming has long been recognized as a vital modelling approach to solve optimization problems [208]. Other approaches for solving optimization problems focus on finding an acceptable, rather than an optimal, solution. This is because for complex optimization problems finding the optimal solution is time-consuming. Examples for these methods include Genetic Algorithms, Memetic Algorithms and Ant Colony Optimization [209].

In this section, the formal mathematical model for minimising the cost of executing BoT jobs whilst ensuring that the users' constraints are satisfied is presented. Mixed Integer Programming (MIP) problems, which are a class of liner programming problems, is used for the modelling.

A linear programming problem (LP) is a mathematical method for determining a way of finding a set of values for continues variables (x_1, x_2, …, x_n) which maximise or minimise a linear objection function z, whilst satisfying a set of constraints. An integer programming problem is a linear programming problem whereby at least one of the variables is restricted to an integer value. If all the variables are restricted to integer values, the model is then known as pure integer programming problem, otherwise it is called mixed integer programming problem [210].

The MIP is a good way of modelling BoT jobs scheduling. The problem is to minimise the costs of executing the BoT job; this can be expressed as the objective function in the MIP. The user constraints, along with the resources available, can be expressed as the constraints functions in the MIP. The scheduling of BoT jobs in the Grid environments is an MIP rather than an LP as a single task within a BoT job is not permitted to be divided into smaller tasks and subsequently allocated to different

resources without any overhead; therefore, a task is only allocated to a single resource for the execution.

The first stage is to define the parameters of the problem and the variables used in the model. We therefore assume we have a BoT job, which has *e* tasks, and a resource provider with *n* resources. The BoT job and the resources parameter are as follows:

t_{jk} total execution time for the *k*th task if assigned to the *j*th resource;

c_j the price per time unit for the *j*th resource;

A_j the time where the *j*th resource is available;

U_j the time where the *j*th resource is unavailable;

R_j ROF of the *j*th resource;

O arrival time of the BoT job;

D user deadline constraint which is a time and date in the future; and

JR user ROF constraint which is the desired ROF level.

The processing time of a task on a resource (i.e. t_{jk}) is assumed to be known. This assumption is a widespread assumption when developing scheduling algorithms in the Grid environments, and this approach is already used by [116, 117, 119, 122, 211]. The reason behind this assumption is the existence of techniques to estimate the task execution time on a given resource. (For more information see 2.6.2.2 Predicting Execution Time).

The resource provider is responsible for setting the price per time unit for the resources. Setting the price of resources is complicated, and some models have been suggested, such as auction or community based models. (The pricing of resources is outside the scope of this thesis and for more information on the subject the reader is referred to [92, 212, 213]). It is noteworthy to highlight that there is no need for the pricing model to be visible to the Grid user [214].

In this chapter, the resources prices are assumed to be a function of performance and ROF, where the higher the resource performance (in terms of processing ability), the higher the resource price per time unit; on the other hand, the higher the resource ROF, the lower the resource price per time unit.

Resources available (unavailable) times are known to the scheduler through the use of advance reservation. The resources ROF are computed with the use of the technique proposed in the previous chapter. The BoT job arrival time is the time at which the job is submitted to the scheduler. The deadline and ROF constraints are the user requirements specified in the SLA.

The variables used in the model are as follows:

x_{jk} = 1 if the *k*th task is assigned to the *j*th resource, otherwise 0 $\forall\, j, k$

s_k start time of the *k*th task $\forall\, k$

y_{jkl} = 1 if the *l*th assignment on the *j*th resource is the *k*th task, otherwise 0 $\forall\, j, k$

f_{jl} the start time of the *l*th assignment on the *j*th resource $\forall\, j, l$

As stated earlier, the allocation of tasks, within a BoT job, to the suitable resources at the appropriate time should be achieved so that the cost of executing the BoT job is minimised and the user requirements are satisfied. This minimisation problem is modelled by the following MIP:

$Minimize$:

$$\sum_{k} t_{jk} \times c_j \times x_{jk} \qquad \forall\, j,k \qquad (1)$$

$Subject\ to$:

$$\sum_{j} O + \left(t_{jk} \times x_{jk}\right) \leq D \qquad \forall\, j,k \qquad (2)$$

$$\sum_{j} R_j \times x_{jk} \leq JR \qquad \forall\, j,k \qquad (3)$$

$$\sum_{j} x_{jk} = 1 \qquad \forall\, j \qquad (4)$$

$$s_k \geq A_j \times x_{jk} \qquad \forall\, j,k \qquad (5)$$

$$s_k + \sum_{j} t_{jk} \times x_{jk} \leq \sum_{j} U_j \times x_{jk} \qquad \forall\, j,k \qquad (6)$$

$$s_k \geq O \times \sum_{j} x_{jk} \qquad \forall\, j,k \qquad (7)$$

$$\sum_{k} y_{jkl} \leq 1 \qquad \forall j,k \qquad (8)$$

$$\sum_{k} y_{jkl} \geq \sum_{k} y_{jk(l+1)} \qquad \forall j,k \qquad (9)$$

- Equation (1) represents the objective function that will be minimized, which is the cost of executing the BoT job. The cost is computed as the sum of the cost of executing the tasks within the BoT job. The expression ($t_{jk} \times c_j \times x_{jk}$) represents the cost of executing the task only if the *k*th task is assigned to the *j*th resource, hence $x_{jk} = 1$. Otherwise, the expression = 0 since $x_{jk} = 0$;
- Equation (2) is the deadline constraint, which all the tasks must finish executing on or before it passes. This constraint ensures that all the tasks assigned to an individual resource *J* finish executing on or before the deadline. This is computed by adding the BoT arrival time and the tasks execution time;
- Equation 3 is the ROF constraint in which the tasks are only assigned to resources with ROF which is less than or equal to the user desired ROF. This constraint ensures that, if a resource is used to execute a task, the resource ROF then does not violate the user ROF requirement;
- Equation 4 ensures that a task is only assigned to one resource, and that all tasks are assigned to resources; this is achieved by ensuring that, for any task, the variable '$x_{jk} = 1$' is obtained for one resource only;
- Equation 5 ensures that the execution of a task on a resource starts only after the resource is available. This is achieved by ensuring that, if a task is assigned to a resource, the time the task starts executing is then after the resource becomes available;
- Equation 6, on the other hand, ensures that the execution of a task is completed before the resource becomes unavailable;
- Equation (7) ensures that the execution of a BoT job only starts after the arrival of the job;

- Equation 8 ensures that there is, at most, one task assigned to a resource at any given time; and
- Equation 9 ensures that a task is assigned to a resource as soon as possible.

6.3.1 Optimal Solution

Solving a LP problem (or MIP) to optimality is complicated. Various different methods have been proposed in the past for solving such problems; these methods include—but are not limited to—the simplex method with its variations, the primal simplex method, the dual simplex method, the interior point method, and the branch and cut method [210, 215, 216]. Importantly, LP is a powerful modelling technique which is used to describe a large number of problems in a number of different fields. For example, LP are used in modelling most of the problems in the operations research community; network and multi-commodity flow problems; the microeconomics and company management, such as planning, production, transportation and likewise; commercial organisations, especially in the current economic climate, which are seeking to maximise profits and minimise costs with limited resources. Owing to the widespread uses of LP, the solving methods aforementioned have been implemented as off-the-shelf software tools, commonly known as solvers. Solvers functionalities differ between different solvers; some only implement a single method and are limited to solving LP problems, whereas others are capable of solving LP and MIP problems. Examples of solvers capable of solving LP and MIP problems include IBM ILOG CPLEX Optimiser [217], Gurobi optimizer [218] and GNU Linear Programming Kit (GLPK) [219]. Another off-the-shelf software tool for LP is the Modelling Language for Mathematical Programming (AMPL) [220]. AMPL is a comprehensive and powerful algebraic modelling language which attains a very high level of readability, since a model written in AMPL resembles the algebraic notation used to formulate LP or MIP problems. Moreover, AMPL is not a solver in itself but rather communicates with different solvers (such as CPLEX or Gurobi) in order to establish a solution for the model.

In order to determine the optimal solution for minimising the costs of executing BoT jobs whilst ensuring that user constraints are satisfied, the MIP is solved using a MIP solver. However, because the scheduling problem is strongly

NP-hard, the solver will not determine the optimal solution in a reasonable amount of time—especially when the size of the BoT job is large or the number of resources available is high. Therefore, an efficient scheduling algorithm for the cost minimisation problem is proposed, known as Deadline and Risk of Failure Constraints algorithm (DRFC).

6.4 The DRFC Algorithm

In the DRFC algorithm, the interest is directed to striking a balance between the objective function and the constraints in order to reduce the BoT execution costs. Therefore, tasks should be allocated to the cheapest suitable resources whenever possible. The cost per time unit does not reflect the true cost of processing, especially when resources have different processing abilities; therefore, the DRFC algorithm will start by calculating the true processing cost for each resource. This is defined as the resource processing ability, and is measured in million instructions per second (MIPS) and divided by the resource price/time unit.

$$True\ Processing\ Cost = \frac{Resource\ Processing\ Ability\ (MIPS)}{Resource\ Cost\ per\ Time\ Unit}$$

The DRFC algorithm sorts the resources in decreasing order, based on the true cost of processing. It is clear that tasks cannot be assigned to resources with ROF higher than the user desired ROF level; therefore, such resources are removed from the sorted list.

The next step in the DRFC algorithm is to arrange the tasks, within a single BoT job, in decreasing order, based on executing time, to be assigned to resources. Starting from the first task in the sorted tasks list, the task needs to be assigned to the first resource in the resources list, if feasible, based on the values of $t_{jk,}$ $A_{j,}$ U_j and D. Subsequently, the task is then removed from the tasks list and the resource variables are updated accordingly. If the task cannot be assigned to the resource, it can be kept within the list, at which point the next task can be considered and the assignment repeated. Once the DRFC goes through the entire tasks list, if there are tasks in the tasks list, then go to the next resource in the resources list, start from the beginning, and repeat the process. This is repeated until the tasks list is empty and a schedule is found or the resources list is empty, before the tasks list, and the BoT job is rejected.

Figure 63 shows the pseudocode for the DRFC algorithm.

// The number of tasks in the BoT job is *e*

// The number of resources in the resource provider domain is *n*

// The MIP parameters are used in the pseudocode

Step 1: // Compute the true processing cost (TPC) for each resource

for ($Resource_1$ to $Resource_n$)

$$TPC\ Resource_i = \frac{Resource_i\ Processing\ ability\ (MIPS)}{Resource_i\ Cost\ per\ Time\ unit}$$

Step 2: // Sort the resources in decreasing order based on TPC

$$TPC\ Resource_1 \geq TPC\ Resource_2 \geq \cdots \geq TPC\ Resource_n$$

Step 3: Remove all Resources with ROF > *JR*

Step 4: // Sort the tasks in decreasing order based on execution time

$$t_{j1} \geq t_{j2} \geq t_{j3} \geq \cdots \geq t_{je}$$

Step 5: // Assign the tasks to resources

Start from the first Resource in the Resources list (*j = 1*)

Start from the first Task in the Tasks list (*k = 1*)

Total cost = 0

While (the Resources list is in not empty)

{

While (the Tasks list is not empty)

{

if (t_{jk} + A_j <= *D)* then

{

Assign the task to the resource

Remove the task from the Tasks list

Update A_j & U_j

Total cost += t_{jk} * c_j

Move to the next task

}

else

Move to the next task in the Tasks list

}

if (the Tasks list is empty) then

Break

else

Move to the next Resource in the Resource list

}

```
if (the Resource list is empty) then
    The BoT job cannot be assigned and therefore rejected
else
    {
    The assignment for the BoT job is found
    The cost of executing the BoT job is Total cost
    }
```

Figure 63: The DRFC Algorithm.

The approach applied to assign tasks to resources—known as the greedy approach—has a number of advantages over other approaches. For example, the algorithms in [117, 211] assign the tasks to resources in the order in which they appear in the BoT job. This approach is not efficient for two reasons: firstly, it is inconsistent and a BoT job—scheduled on the same resources—will have different assignments if the order of tasks in the BoT job is changed; and secondly, it does not fully utilise the resources, and a BoT job might be rejected, although an assignment is feasible. In order to illustrate these limitations, a simple example is given.

Assume there are two resources with the same processing ability, and a BoT job is submitted for processing with 100 time units as a deadline. Both resources are suitable for executing the tasks; Resource A is available from the time the BoT job is submitted, whilst Resource B is available after 50 time units. Resource A is cheaper than Resource B; thus, tasks will be assigned to Resource A first. Let's assume that the time the BoT job was submitted is 0. The BoT job has three tasks, which are to be carried out in the following order:

1. Task 1 execution time is 30 time unit.
2. Task 2 execution time is 40 time unit.
3. Task 3 execution time is 70 time unit.

Assigning the tasks to resources in the order in which they appear will result in the rejection of the BoT job. Figure 64 shows that, after Task 1 and 2 are assigned, Resource A's available time is 30 time units and Resource B's available time is 50 time units. Both resources are not able to execute Task 3, which requires 70 time units.

The above assignment approach rejects the BoT job, despite there being two possible schedules. Accordingly, Task 1 and 3 should be assigned to Resource A,

and Task 2 to Resource B (Figure 65); otherwise, Task 2 and 3 should be assigned to Resource A and Task 1 to Resource B (Figure 66). The latter is better as the cheaper resource is used for a longer period. Essentially, using the greedy approach will result in the latter assignment, and will always be consistent regardless of the tasks order.

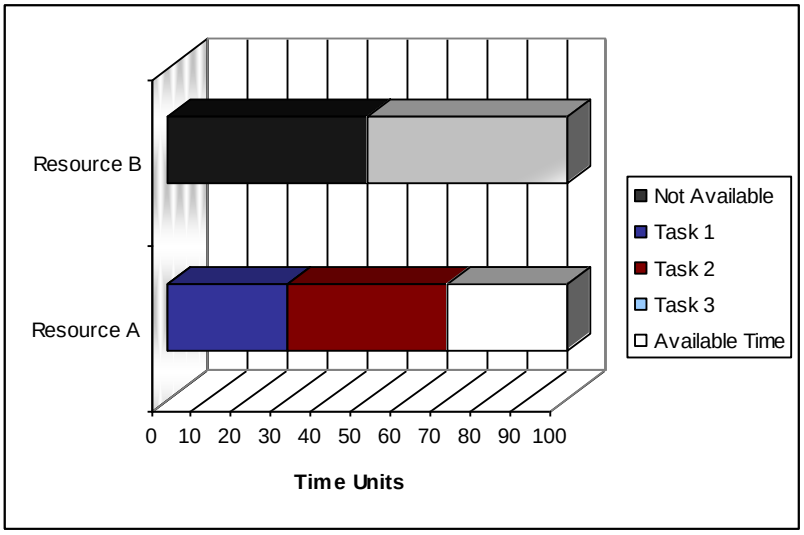


Figure 64: Assignment of Task 1 & 2 to Resource A.

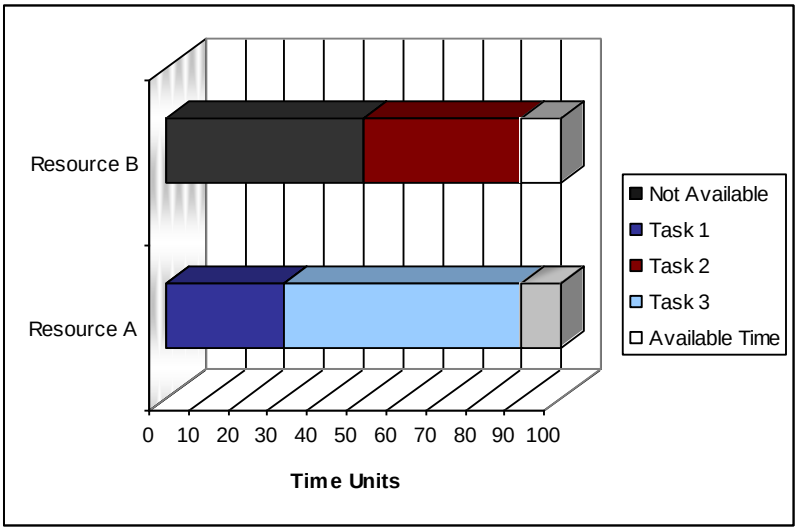


Figure 65: Assignment of Task 1 & 3 to Resource A and Task 2 to Resource B.

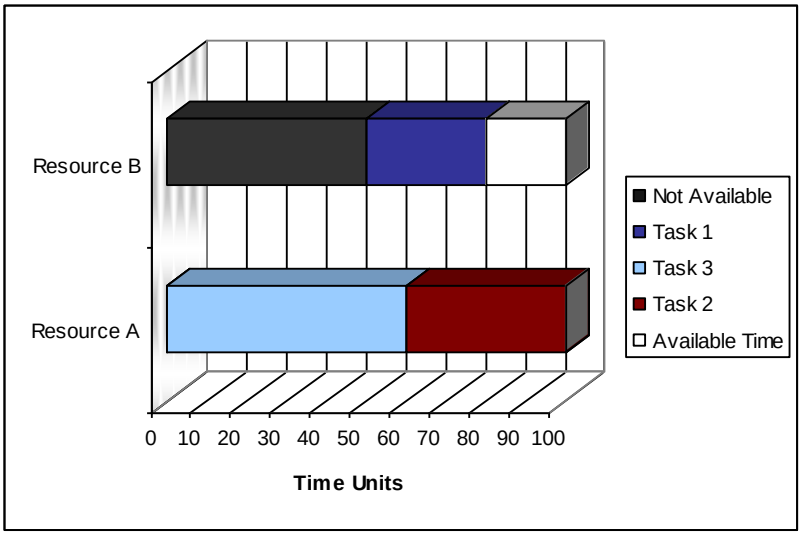


Figure 66: Assignment of Task 2 & 3 to Resource A and Task 1 to Resource B.

6.5 Simulation-Based Performance Analysis

A variety of methods exist for carrying out performance evaluation of resource scheduling algorithms. Some evaluation methods include: analytical, simulation, emulation and empirical. Yet the DRFC algorithm is evaluated using simulation. This is because simulation has a number of advantages

- the ability to conduct experiments in controlled environments;
- the potential to obtain insight concerning the interaction of the experiment variables;
- the potential to obtain insight regarding the effect of changing a single variable whilst others are fixed; and
- no limits to experimental scenarios, which makes it possible to reproduce the results [221].

The purpose of the simulation is to test the allocation of tasks in a BoT job with the use of the DRFC algorithm, and to accordingly compare it with the optimal allocation obtained by solving the MIP problem using the Gurobi optimiser 4.0 [218]. The simulation environment and variables should be identical to the MIP solver environment; thus, the differences in experiments results are not owing to the environment.

There are a number of tools for simulating Grid computing environments such as GridSim [118], SimGrid [222] and Optorsim [223]. However, these tools do not have a method to represent the resource ROF and they do not interact with MIP solvers. Therefore, the discrete event simulation tool used in the experiments has been developed form scratch. Both the DRFC algorithm simulator and the solver for the MIP are written in C++ VisualStudio 2008. The event scheduling approach is used in the simulation were the events, arrival of BoT job, might change the status of resources if an allocation is found. In the remainder of this section the experiments design is presented along with resource modelling and workload modelling. This is followed by the experiments results and the sensitivity analysis of the deadline and the ROF constraints.

6.5.1 Experiments Design

In order to test the DRFC algorithm in different settings, different resource providers and different workloads, three resource providers are considered:

1. Resource provider 1 with 10 resources.
2. Resource provider 2 with 50 resources.
3. Resource provider 3 with 100 resources.

For each resource provider three scenarios are considered.

1. The BoT jobs submitted have less than 50 tasks.
2. The BoT jobs submitted have less than 100 tasks.
3. The BoT jobs submitted have less than 200 tasks.

The number of resources in the resource provider was selected from 10 to 100 so as to enable the Gurobi optimiser to determine a solution for the MIP problem, considering hundreds or thousands of resources make finding a solution for the MIP problem unfeasible. The average BoT size—submitted to the Grid systems in the real world—is between 5 and 50 tasks [224], hence the selection of 50 and 100 tasks for the BoT size. The 200 tasks size was selected to investigate the performance of the DRFC algorithm when the BoT Job is large.

In the rest of this chapter, Resource Provider 1 is referred to as a small provider, Provider 2 as a medium provider, and Provider 3 as large provider. Furthermore, the BoT Job 1 is referred to as small size job; BoT Job 2 as a medium job, and BoT Job 3 as a large job. This naming is only used in order to simplify the writing and to facilitate reader understanding in relation to the resource providers and BoT jobs.

In order to carry out the simulation, the resource providers and the workloads need to be modelled. The following two subsections represent the modelling of resource providers and BoT jobs workload.

6.5.1.1 Resource Provider Modelling

Resources within a resource provider domain are not identical; they have different characteristics, configurations and capabilities. For these experiments the

resources are modelled using three metrics: the processing ability measured in million instructions per second (MIPS), price per time unit, and ROF.

In this simulation the processing ability of resources is assumed to be between 4000 MIPS and 8000 MIPS in steps of 1000. The price of a resource per time unit is assumed to be from 0.6 to 1.8 units. The ROF of a resource is < 0.05, ≤ 0.1 or > 0.1. Table 14 shows the exact parameters used for resources in the experiments.

Table 14: Resources Used for the Simulation.

MIPS	*ROF*	*Price*	*ROF*	*Price*	*ROF*	*Price*
4000	< 0.05	1	≤ 0.1	0.8	> 0.1	0.6
5000	< 0.05	1.2	≤ 0.1	1	> 0.1	0.8
6000	< 0.05	1.4	≤ 0.1	1.2	> 0.1	1
7000	< 0.05	1.6	≤ 0.1	1.4	> 0.1	1.2
8000	< 0.05	1.8	≤ 0.1	1.6	> 0.1	1.4

The resource price per time unit was randomly assigned, yet two conditions were considered.

1. If two resources have the same ROF, then the resource with the lower processing ability is always cheaper; and
2. Two resources with different processing abilities might have the same price if they have a different ROF.

In the real world, the resource ROF changes with time; however, in this simulation, it is assumed to remain constant in order to make the resources identical throughout the simulation. Another note regarding the ROF is that it is not limited to the three values assumed, but it is the responsibility of the resource provider to explicitly specify the type of resources available, the ROF of those resources, and the price per time unit.

The processing ability is assumed to be in the range 4000 to 8000 MIPS. The grounds for this assumption is taken from Buyya *et al*. [211] who, in 2002, provided different MIPS for different resources and the average MIPS was around 400 MIPS. In 2010, using Moore's Law [225], the average should be around 6400. Therefore, 6000 MIPS is considered to be the average processing ability in the simulation.

Another reason for this assumption is that resources processing ability is used in modelling BoT jobs. Thus the change of resources MIPS don't change the processing time of jobs and simulation results will be similar in both cases. This will be clarified in the next subsection.

6.5.1.2 Workload Modelling

The BoT workload can be either a real workload (trace) or a workload driven from a model. Both have advantages and disadvantages. The advantage of using a trace directly is that it is the most 'real' test of the Grid system under study, and the workload reflects a real workload precisely, with all its complexities. The drawback is that the trace reflects a specific workload, and there is always the question of whether or not the results can be generalised to other Grid systems or load conditions. Workload models have a number of advantages over traces [226].

- It is easy to know which workload parameters are correlated with each other because this information is part of the model;
- It is possible to change model parameters one at a time in order to investigate the influence of each one, whilst keeping other parameters constant;
- A model is not affected by policies and constraints which are particular to the Grid site where a trace was recorded; and
- Traces may be polluted by bogus data.

The BoT workload used for these experiments is based on the realistic workload model for BoT jobs introduced in [224]. Seven Grid workload traces from the Grid Workloads Archive (GWA) [227] were used to validate the model.

In these experiments, the interest are on three aspects: the BoT jobs arrival rate, the BoT jobs sizes (i.e. the number of tasks in the job), and the task characteristics. The BoT jobs arrival rate during peak hours is modelled with a Weibull distribution. The Weibull distribution is also used to model the BoT jobs sizes. The tasks characteristics are the average task run time and the variance of run times of the tasks in a single BoT job. The average run time is modelled with a normal distribution, and the variance of run times is modelled by a Weibull distribution. Table 15 shows the exact parameter values for the workload model.

Table 15: The Parameter Values for the Workload Model. W stand for Weibull and N for Normal distribution [224].

BoT Arrival	*BoT Size*	*Task Average Run Time*	*Task Variance*
W(4.25, 789)	W(1.79,24.16)	N(2.73,6.1)	W(2.05, 12.25)

The Matlab environment [228] was used to generate the workload values based on the distributions. For each BoT job in the workload, the arrival time is recorded. The average task runs time with the tasks run time variance and size, for each BoT job, is used to compute the individual tasks run time. The task length in machine instruction (MI) is computed using the formula:

$$\text{Task Length} = \text{Task Run Time} \times \text{Resource Processing ability (MIPS)}$$

Where the resource processing ability is assumed to be the average MIPS for the simulation. Recall, in the last subsection, it was indicated that the selection of resources processing abilities will not affect the simulation as the workload is based on the task run time—regardless of the processing ability.

6.5.2 Simulation Results

There are, in total, nine experiments: small, medium and large resource providers each has three scenarios to consider in which small, medium or large BoT jobs are submitted to be scheduled into resources. Each experiment is evaluated using two criteria. The first criterion is the percentage of the number of BoT jobs scheduled using the DRFC algorithm and the optimal schedule using Gurobi optimizer. Since finding the optimal schedule is a memory-intensive process it is common that the resource executing the allocation algorithm run out of memory before the BoT jobs are scheduled. The number of BoT jobs scheduled using DRFC algorithm is the base for calculating the percentage; less than 100% indicate that the Gurobi optimiser run out of memory and stop working before finishing the workload, higher than 100% indicate that the Gurobi optimiser scheduled more BoT jobs than the DRFC algorithm and 100% indicate that both scheduled the same number of jobs. The second evaluation criterion is the average difference between the costs of executing a BoT job when scheduled with the DRFC algorithm and the optimal costs of executing the BoT job when scheduled using Gurobi optimizer.

Only the jobs that have been scheduled using DRFC algorithm and Gurobi optimizer are considered in the difference the rest are discarded.

$$Criterion\ 1 = \frac{Number\ of\ Jobs\ Scheduled\ Using\ Gurobi}{Number\ of\ Jobs\ Scheduled\ Using\ DRFC\ Algorithm} \times 100$$

$$Criterion2 = \frac{Execution\ Cost\ using\ DRFC - Optimal\ Execution\ Cost}{Optimal\ Execution\ Cost} \times 100$$

Since different workloads scheduled on different resources provide different results. Each experiment is repeated 10 times with different workloads and different resources and the results shown here are the averages of the 10 experiments. The number of BoT jobs in a workload is set to 100 jobs.

For small workloads the deadline constraint is varied in simulation time between 1000 and 5000 in steps of 1000, for medium workload the deadline constraint is varied between 2000 and 10000 in steps of 2000 and for the large workloads the deadline constraint is varied between 5000 and 25000 in steps of 5000. The deadline is considered from the BoT job arrival time. The ROF constraint is similar to resources ROF (i.e. ROF < 0.05, ROF ≤ 0.1 and ROF > 0.1).

For all the experiments DRFC is coded in C++ Visual Studio 2008 and Gurobi optimizer 4.0 was used with the C++ interface. The machine used on these experiments is Intel core 2 quad CPU Q9300 2.5 GHz and 3 GB RAM. (See Appendix C for the code validation).

6.5.2.1 Summary of Results

Experiment 1- Small Resource Provider Running Small BoT Jobs:

Figure 67 shows that finding the optimal schedule is not effective since there is a high possibility that the resource executing the optimal allocation algorithm ran out of memory. The Gurobi solver, on one instant, assigns slightly more BoT jobs, which is when the deadline constraint is 1000 and the ROF constraint is < 0.05. However, in the rest of the simulation, the DRFC outperforms the Gurobi solver. Figure 68 shows that the average difference in cost between the DRFC and the optimal cost is minimal and in most cases less than 0.8%.

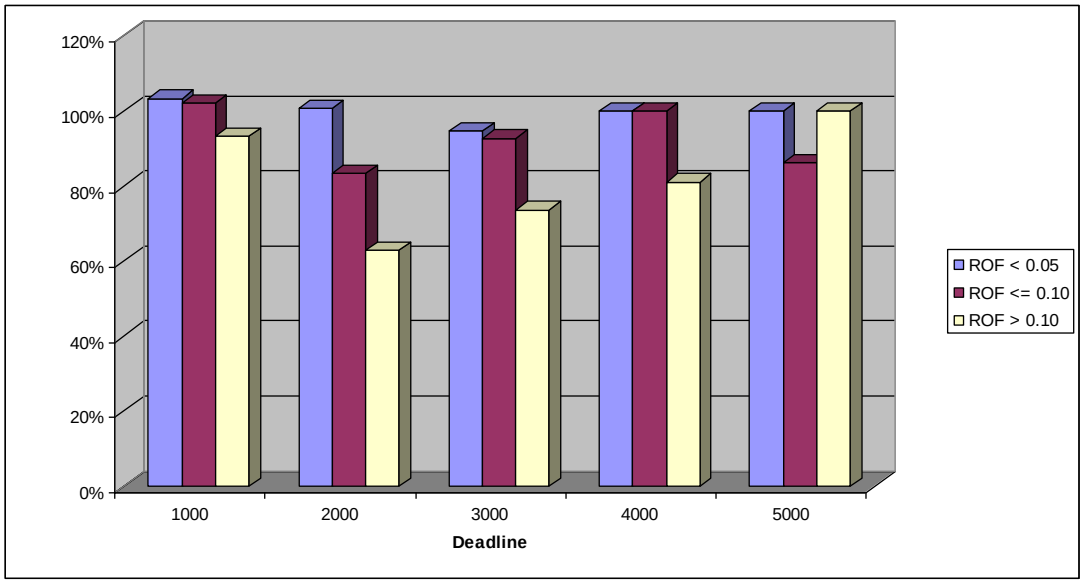


Figure 67: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 1.

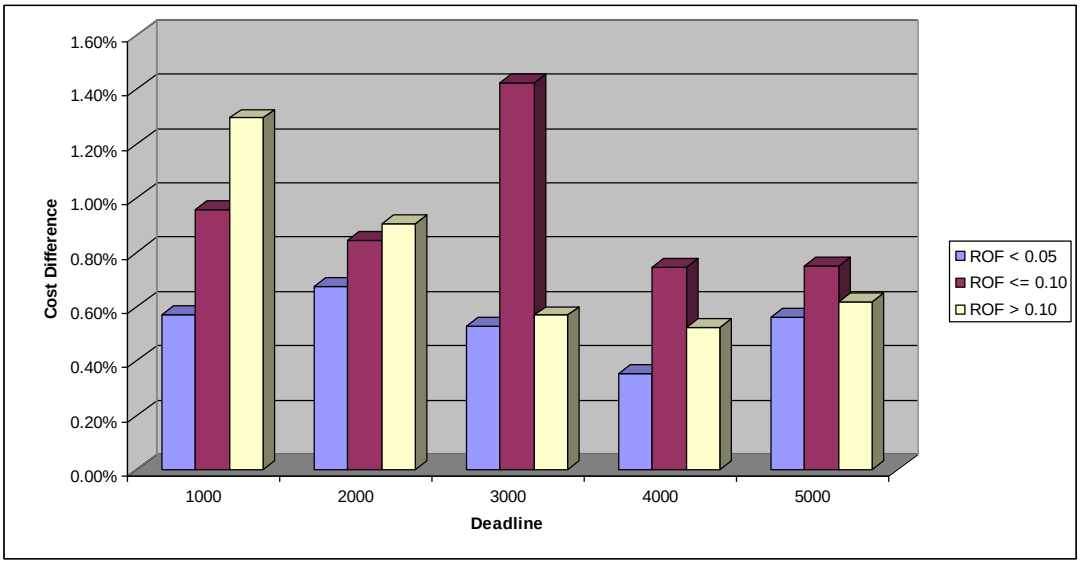


Figure 68: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 1.

Experiment 2- Small Resource Provider Running Medium BoT Jobs:

Figure 69 shows that the optimal allocation algorithm performs badly, particularly when the ROF > 0.01. The reason is that, with ROF > 0.01, all resources in the resource provider domain are considered, which increases the size of the allocation problem, thus requiring more computational time and memory. Figure 70 shows that the average difference in cost between the DRFC and the optimal cost is minimal, with most cases being less than 1%.

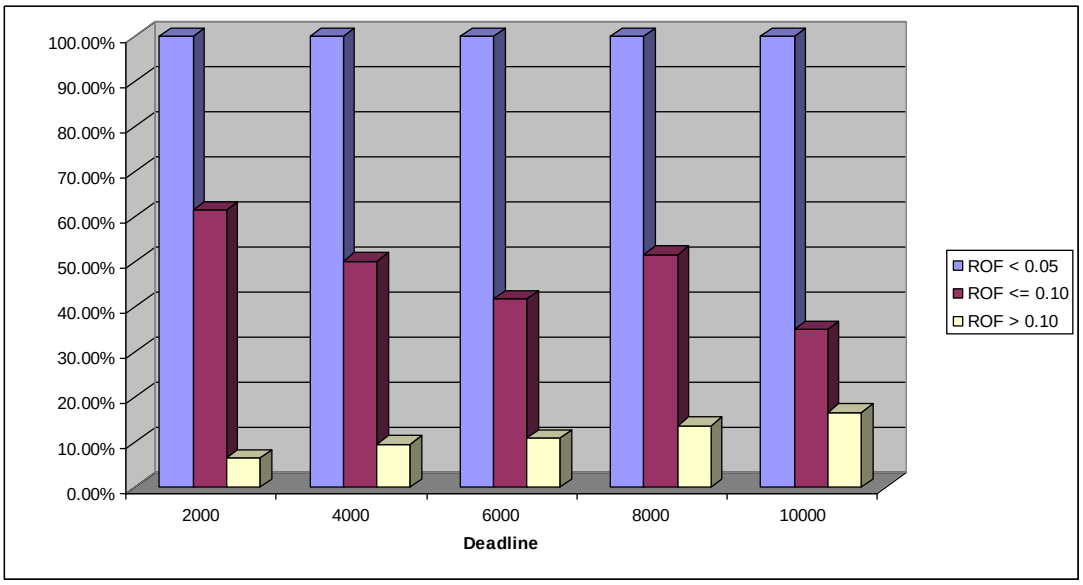


Figure 69: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 2.

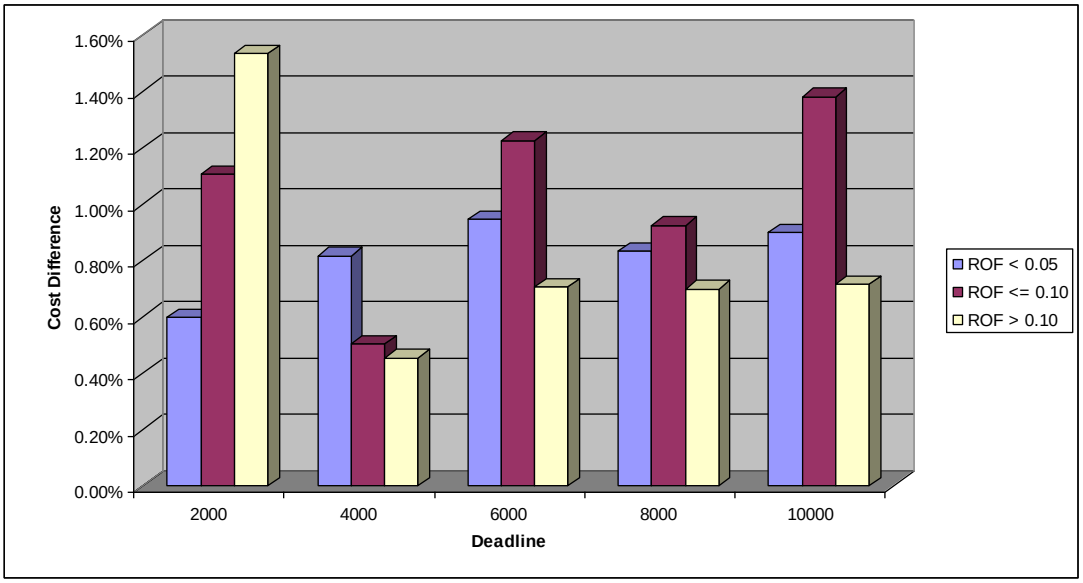


Figure 70: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 2.

Experiment 3- Small Resource Provider Running Large BoT Jobs:

Figure 71 shows that the optimal allocation algorithm performs badly in a similar way to Experiment 2. Figure 72 shows that the average difference in cost between the DRFC and the optimal cost is minimal, with most cases being less than 1.5%.

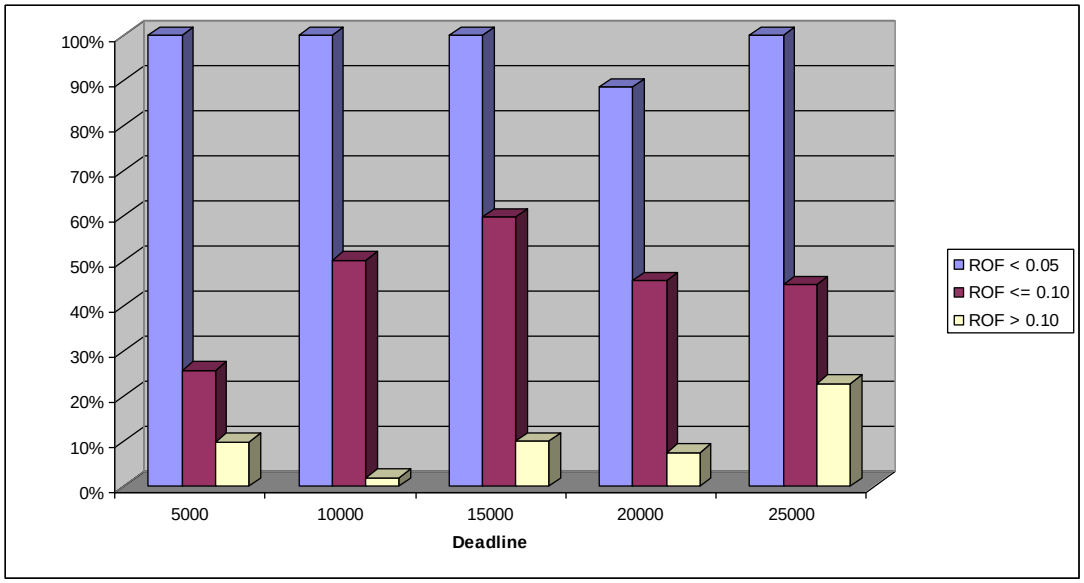


Figure 71: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 3.

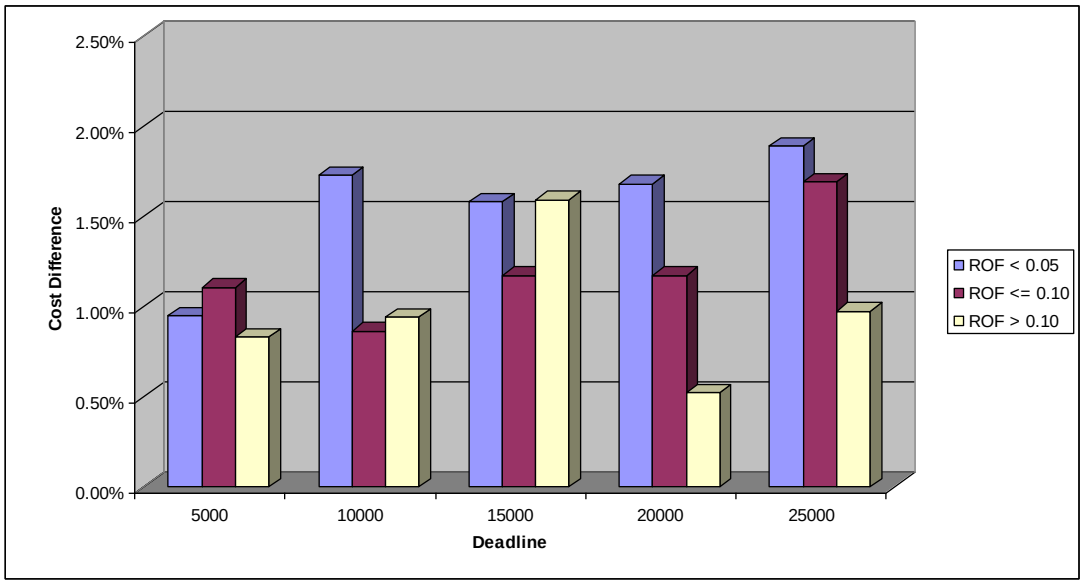


Figure 72: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 3.

Experiment 4- Medium Resource Provider Running Small BoT Jobs:

Figure 73 shows that the optimal allocation algorithm performs badly, especially with short deadline constraint. Figure 74 shows that the average difference in cost is minimal, with most cases being less than 0.6%.

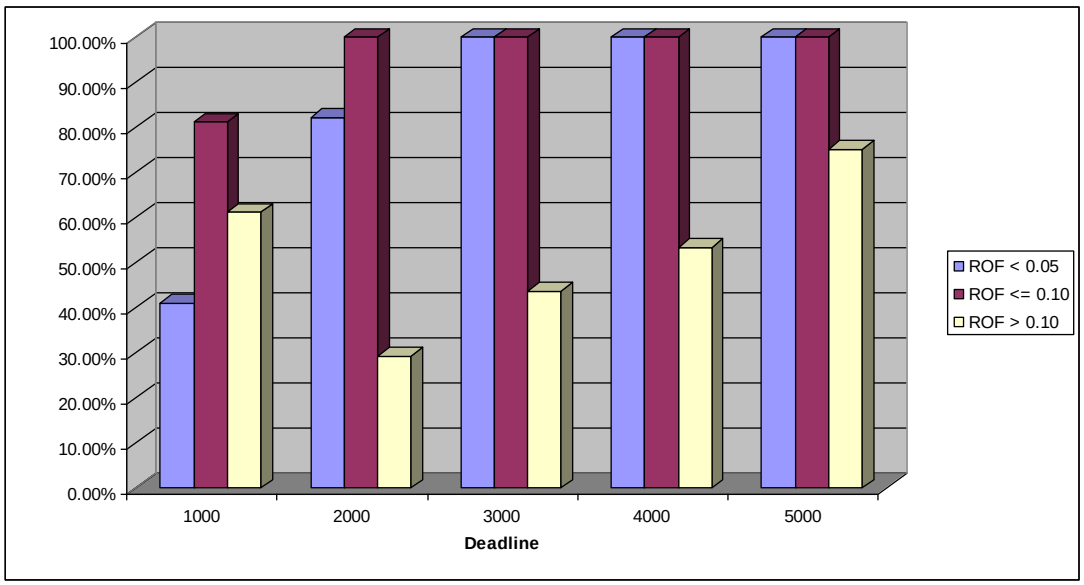


Figure 73: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 4.

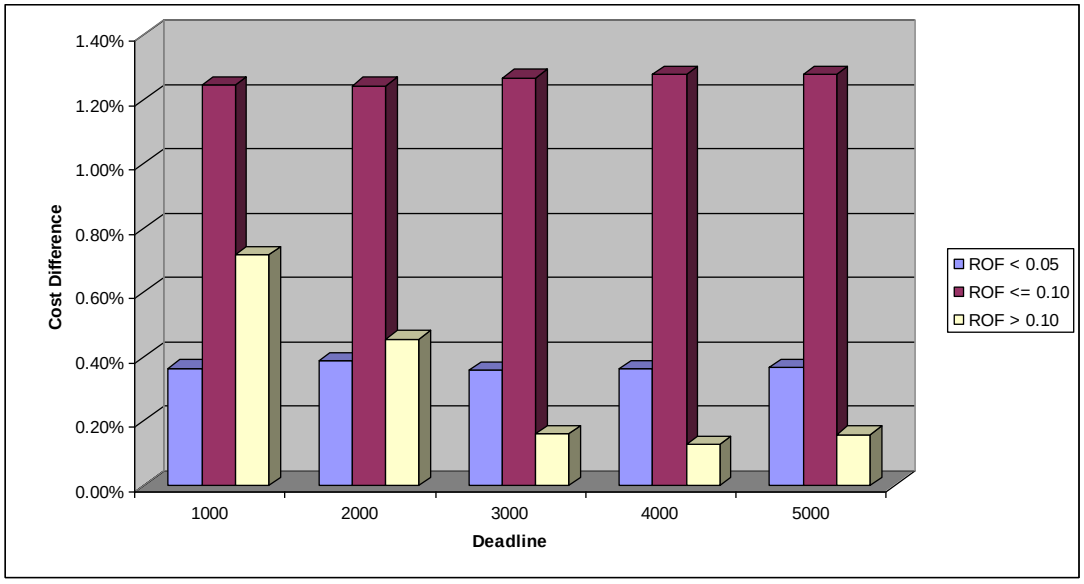


Figure 74: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 4.

Experiment 5- Medium Resource Provider Running Medium BoT Jobs:

Figure 75 shows that the optimal allocation algorithm performs badly in all instances, regardless of the exact constraints. Figure 76 shows that the average difference in cost is minimal, with most cases being less than 1%.

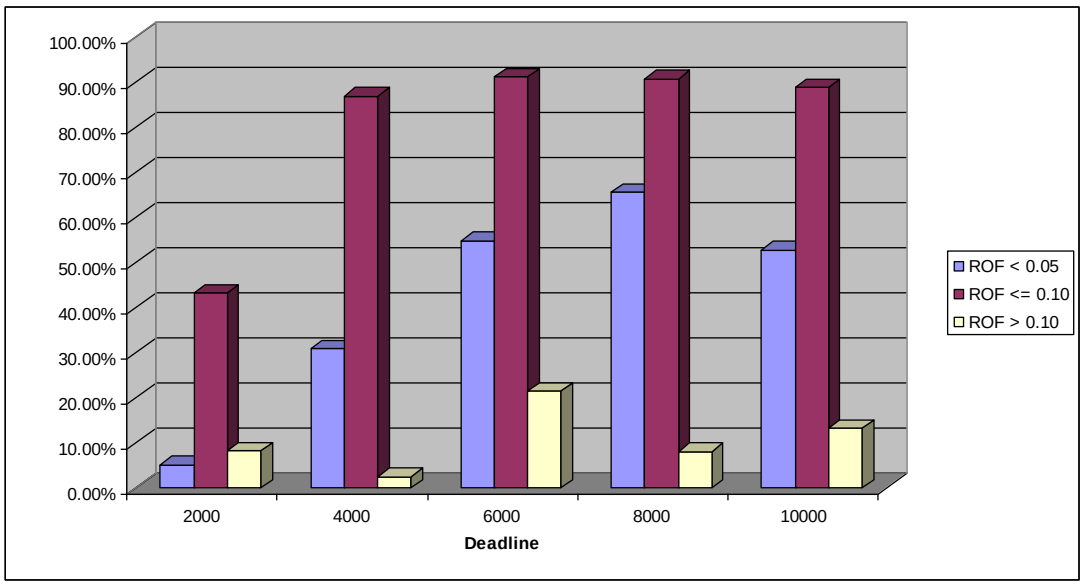


Figure 75: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 5.

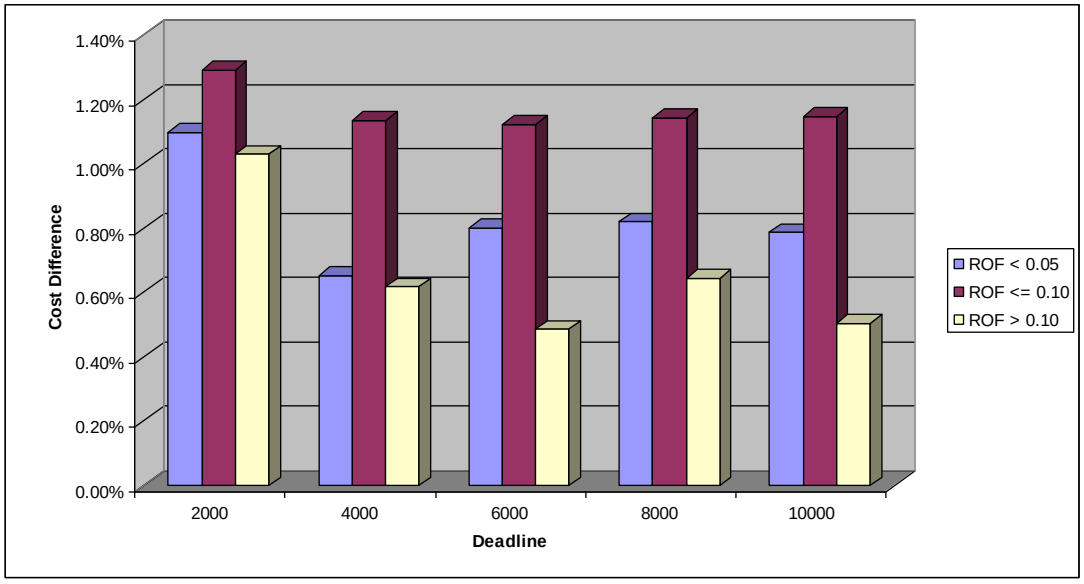


Figure 76: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 5.

Experiment 6- Medium Resource Provider Running Large BoT Jobs:

Figure 77 shows that the optimal allocation algorithm performs extremely badly. The reason for this is that, with large BoT jobs, the allocation problem size increases; hence requiring more computational time and memory. Figure 78 shows that the average difference in cost is minimal, with most cases being less than 0.8%.

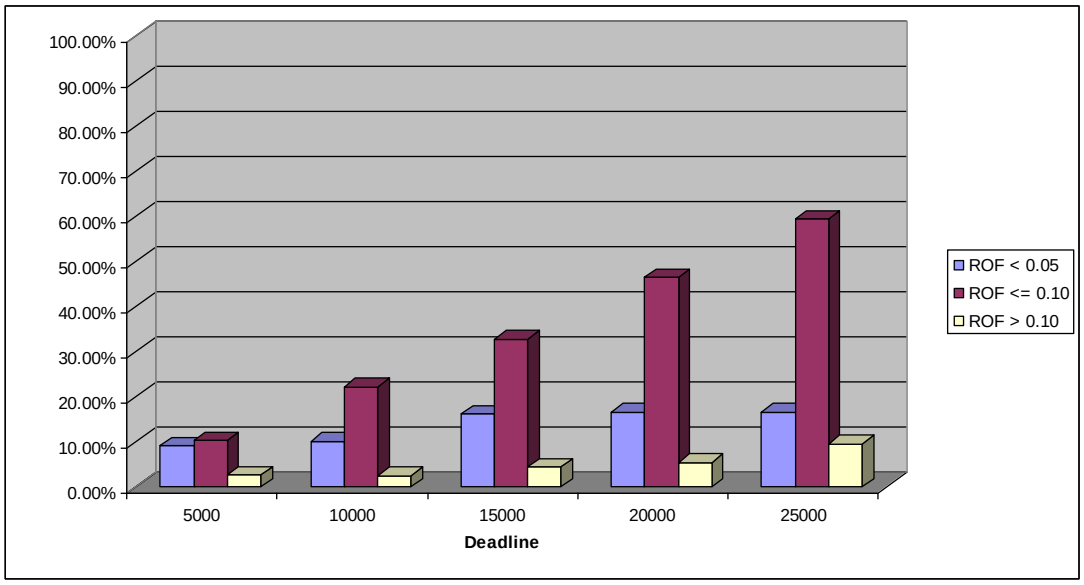


Figure 77: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 6.

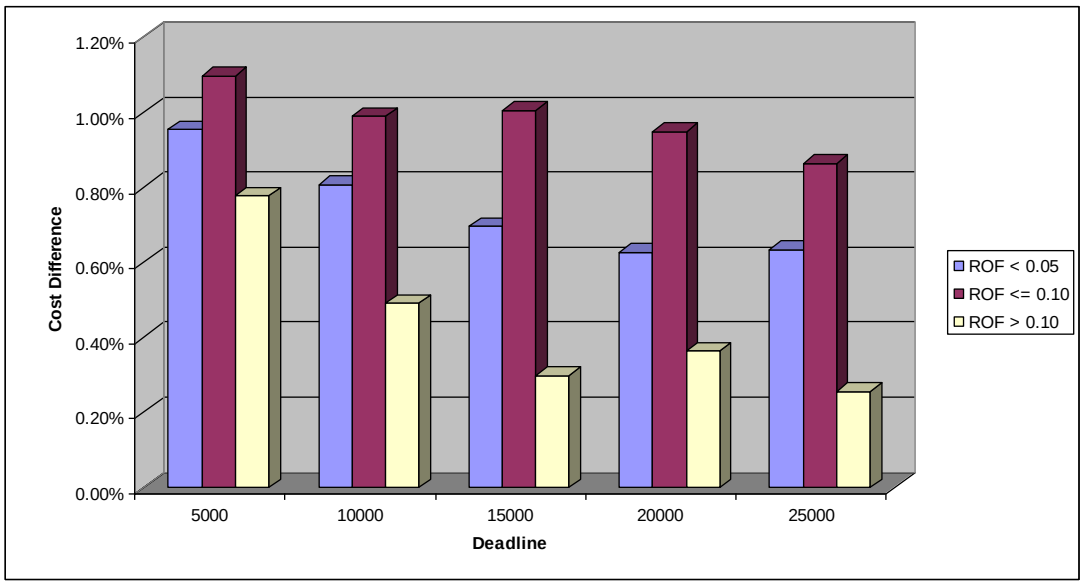


Figure 78: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 6.

Experiment 7- Large Resource Provider Running Small BoT Jobs:

Figure 79 shows that the optimal allocation algorithm performs badly in a similar way as experiment 2. Figure 80 shows that the average difference in cost is minimal, with most cases being less than 0.4%.

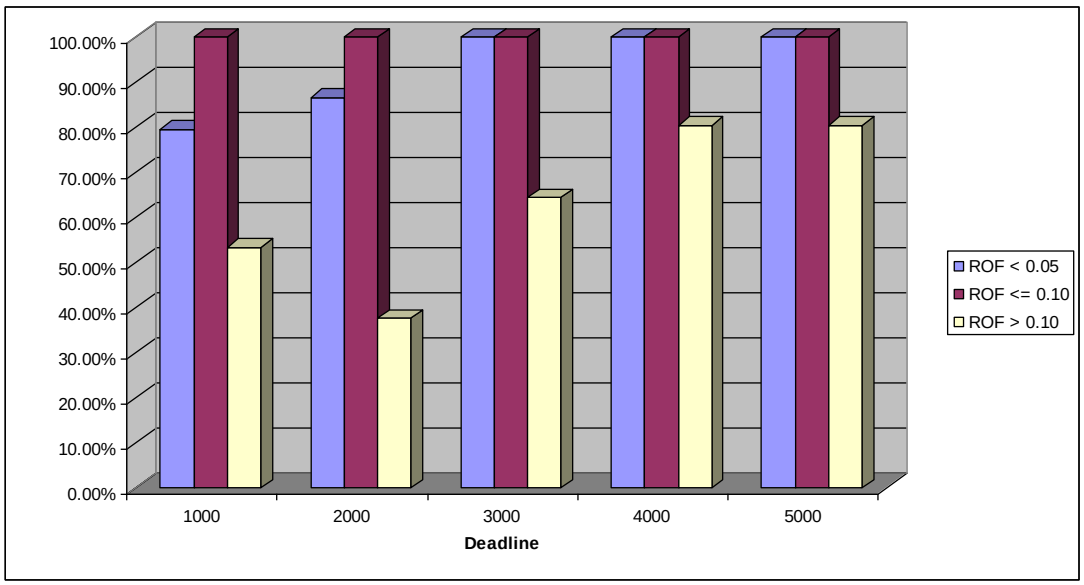


Figure 79: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 7.

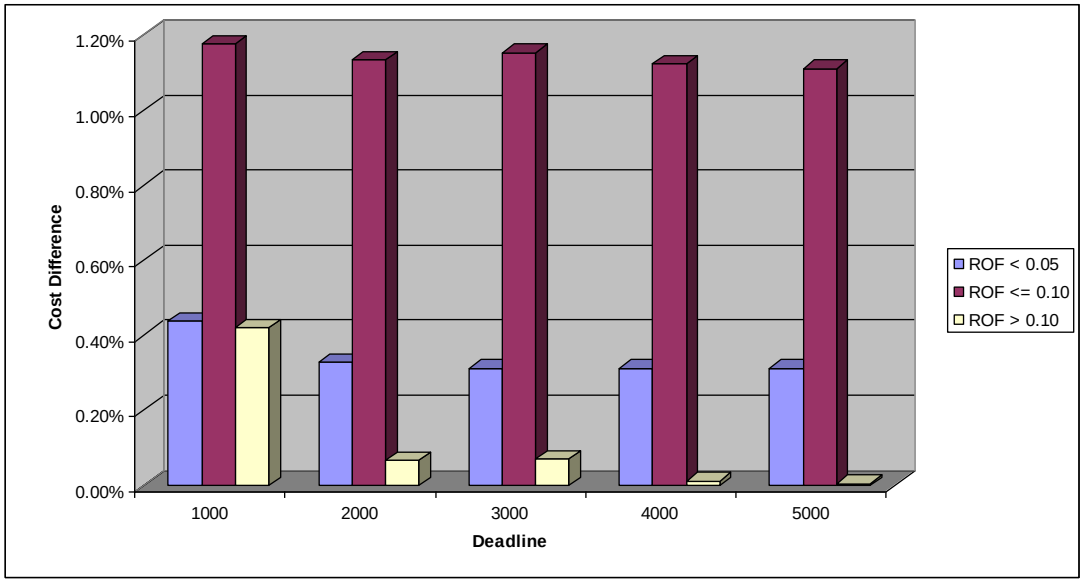


Figure 80: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 7.

Experiment 8- Large Resource Provider Running Medium BoT Jobs:

Figure 81 shows that the optimal allocation algorithm performs badly in a similar way as experiment 2. Figure 82 shows that the average difference in cost is minimal, with most cases being less than 0.6%.

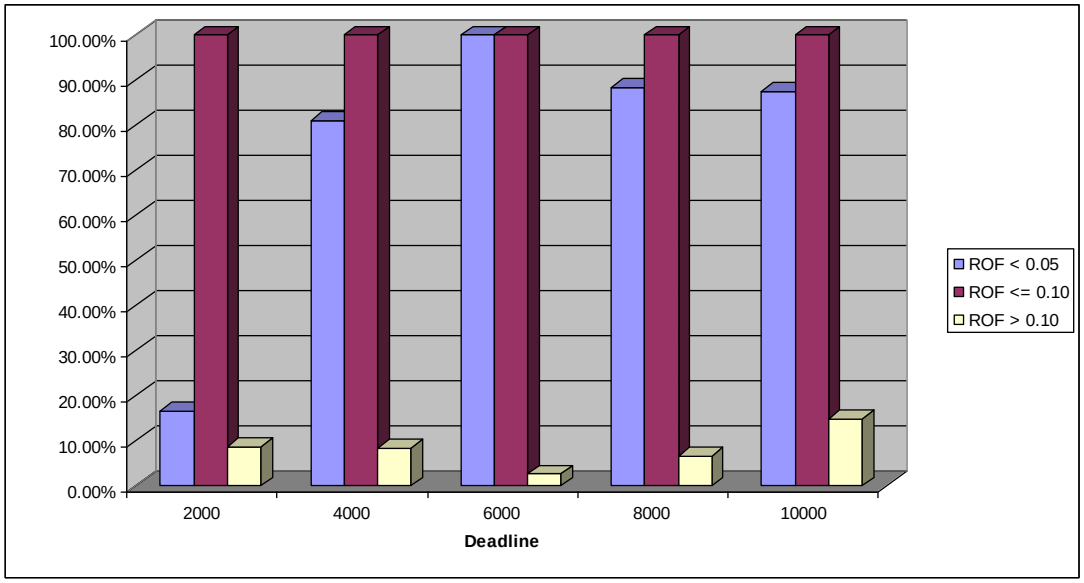


Figure 81: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 8.

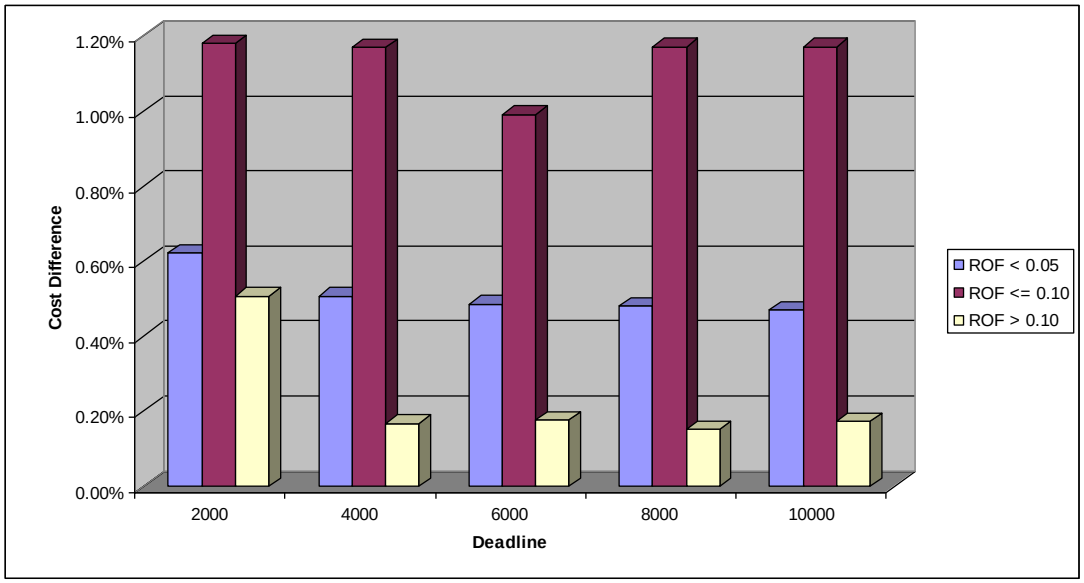


Figure 82: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 8.

Experiment 9- Large Resource Provider Running Large BoT Jobs:

Figure 83 shows that the optimal allocation algorithm performs badly in a similar way as experiment 2. Figure 84 shows that the average difference in cost is minimal, with most cases being less than 0.6%.

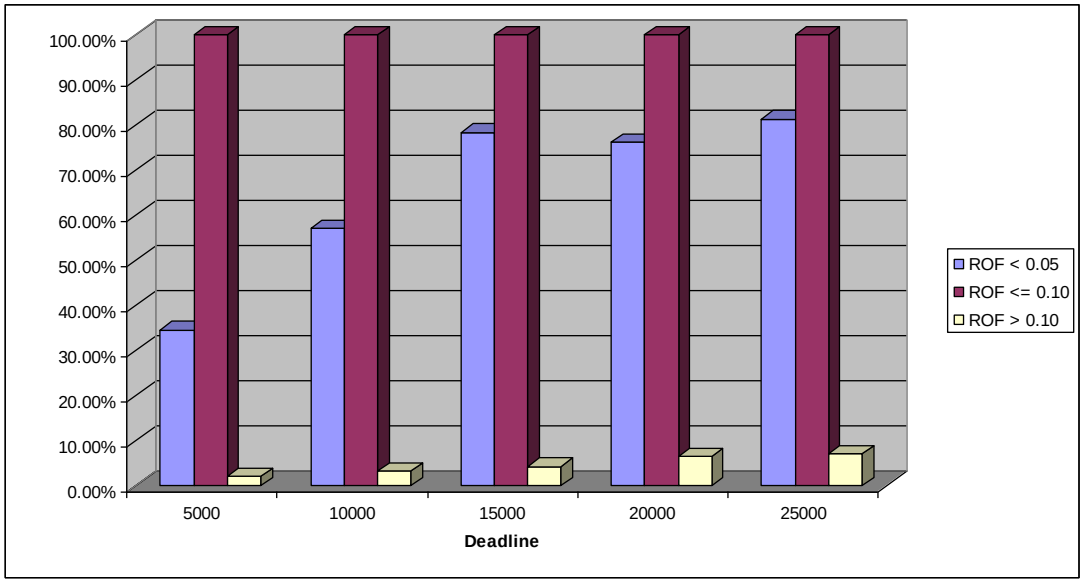


Figure 83: Percentage Difference in Number of Jobs assigned with DRFC & Gurobi Experiment 9.

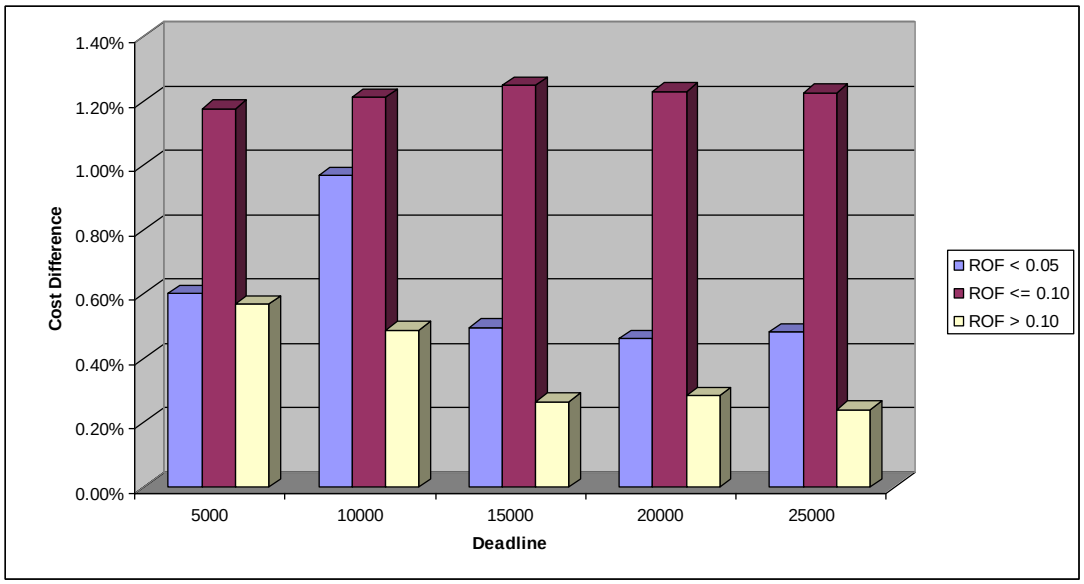


Figure 84: Percentage Difference between DRFC Execution Cost and Optimal Cost Experiment 9.

The experiments show that finding the optimal allocation of tasks is an intensive process, which requires a powerful computer to compute with a huge internal memory; thus, it is inefficient to use this method for real-time scheduling. The DRFC algorithm proposed provides a good solution for a wide variety of resource providers and workloads. The difference in the execution cost between the optimal solution and the solution found using DRFC algorithm is minimal and in

most of the cases less than 1%. Another important feature of the DRFC algorithm is that it takes a short amount of time to determine a scheduler for the BoT job—on average, totalling approximately 1 millisecond—whilst the optimal solution, most of the time, is not found in a reasonable amount of time.

From the above, it can be stated that the DRFC algorithm performs better than the optimal scheduler—especially for large BoT jobs and large resource providers. The cost of executing a BoT job scheduled using the DRFC algorithm is near-optimal. Finally, the time to find a scheduler with DRFC algorithm is minimal; this makes the DRFC algorithm a superior choice for the real-time scheduling of BoT jobs in Grid environments.

6.5.2.2 Sensitivity to The Deadline and The ROF

The effect of the deadline and ROF constraints differ between different resource providers and BoT jobs sizes. When the resources available are not capable of executing all the BoT jobs submitted, either because the number of resources available is small or the size of the BoT jobs is large, the deadline and ROF constraints are responsible for the number of BoT jobs accepted for scheduling and the number of BoT jobs rejected. When the resources available are capable of executing all the BoT jobs submitted, and there aren't any rejections, the deadline and ROF constraints affect the price of executing the BoT jobs. This is further explained below using the results of the DRFC algorithm, since the DRFC algorithm only reject a BoT job if no schedule was found unlike the Gurobi optimiser where it might stop working before the workload is finish.

Figures 85, 86 & 87 show the number of jobs accepted for different deadline and ROF for a small resource provider running small BoT jobs, medium BoT jobs and large BoT jobs respectively. As expected, the number of jobs accepted increase when the deadline is increased and the number also increase when the ROF is increased, even if the deadline is not change. This is because when the ROF increased resources that were available but have higher ROF can be used.

For the BoT job owner (resources user), increasing the deadline and/or the ROF constraints increases the chances that his/her BOT job is accepted by the resource provider. This is especially true when the BoT job is large.

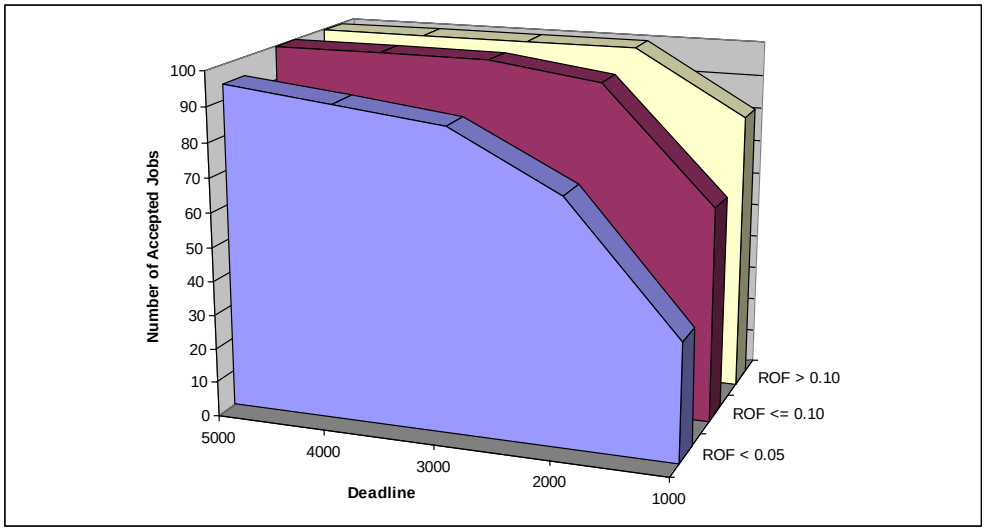


Figure 85: Number of BoT Jobs Accepted, Small Provider Running Small BoT Jobs.

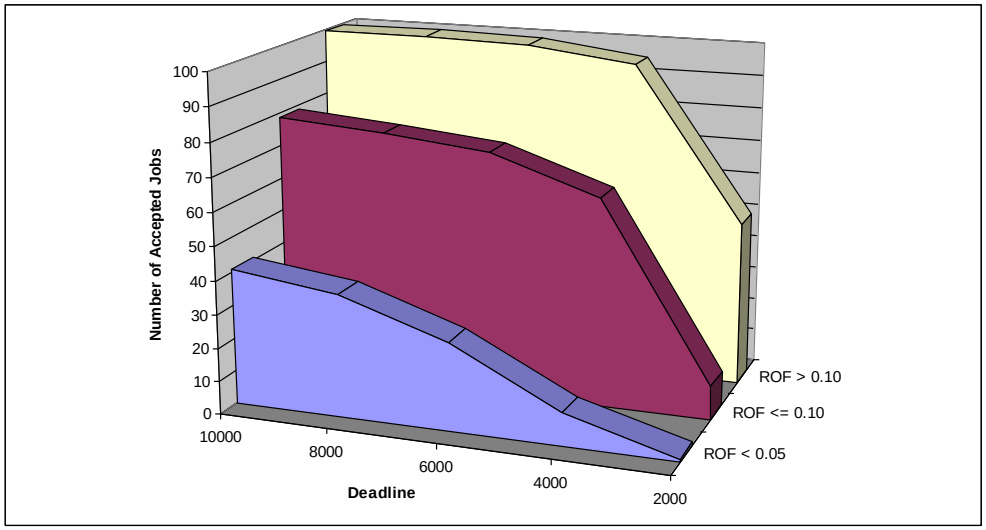


Figure 86: Number of BoT Jobs Accepted, Small Provider Running Medium BoT Jobs.

Figures 88, 89 & 90 show the total cost for executing an entire workload with different deadline and ROF for a large resource provider running small BoT jobs, medium BoT jobs, and large BoT jobs respectively. As expected, executing the BoT jobs on resources with ROF < 0.05 is the most expensive. The cost is decreased when the ROF constraints are relaxed; this is because resources with a higher ROF are cheaper. Increasing the deadline will decrease the total cost, although the drop off is small and after a threshold the total cost will remain constant—even if the deadline is increased. The threshold is when the entire workload is executed only on the cheapest resources.

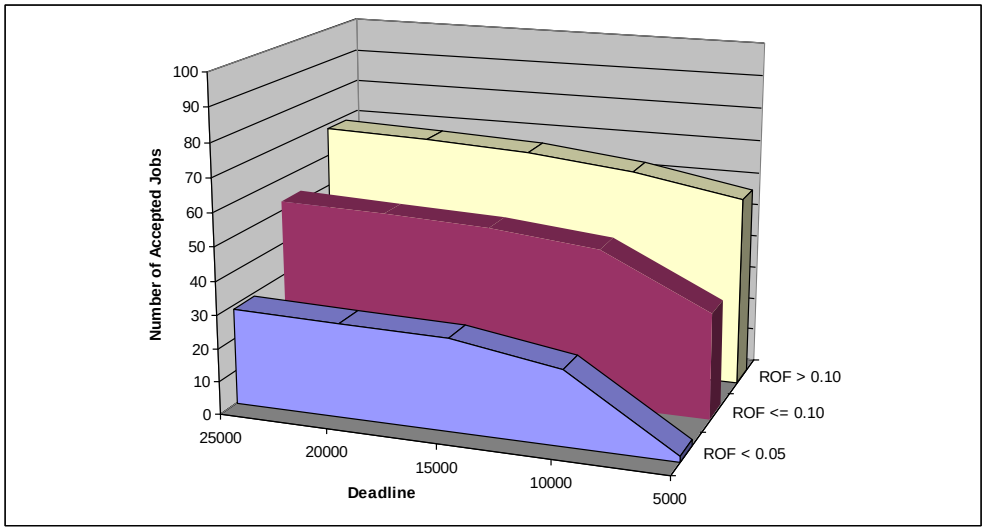


Figure 87: Number of BoT Jobs Accepted, Small Provider Running Large BoT Jobs.

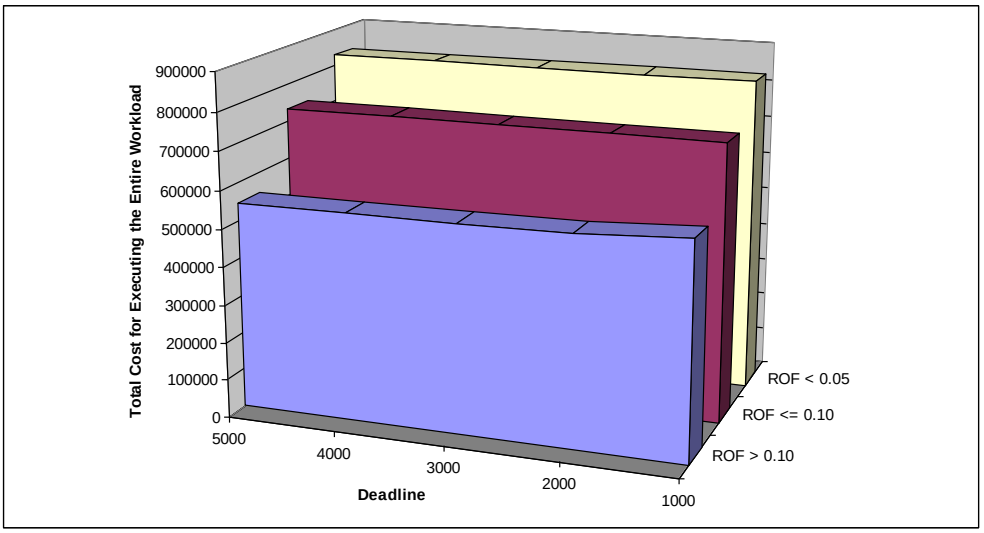


Figure 88: Total Execution Cost, Large Provider Running Small BoT Jobs.

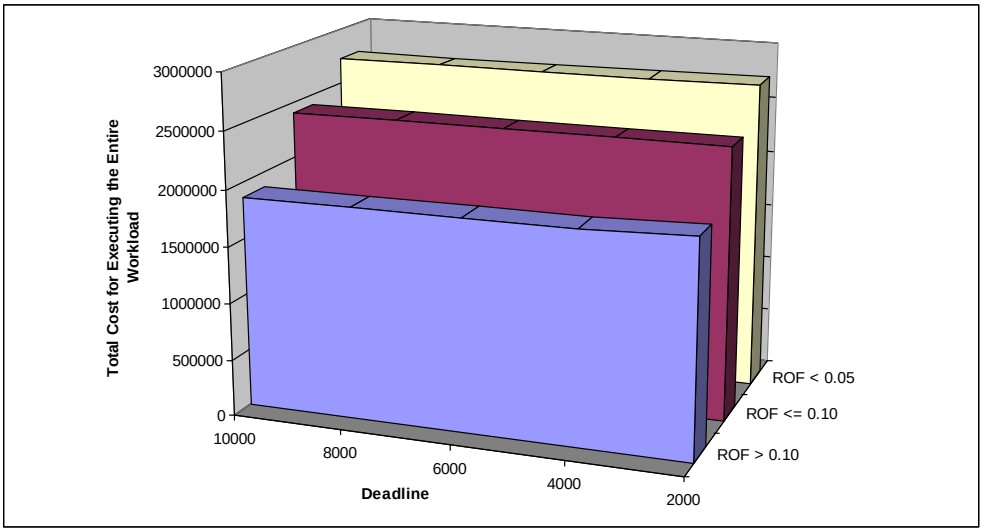


Figure 89: Total Execution Cost, Large Provider Running Medium BoT Jobs.

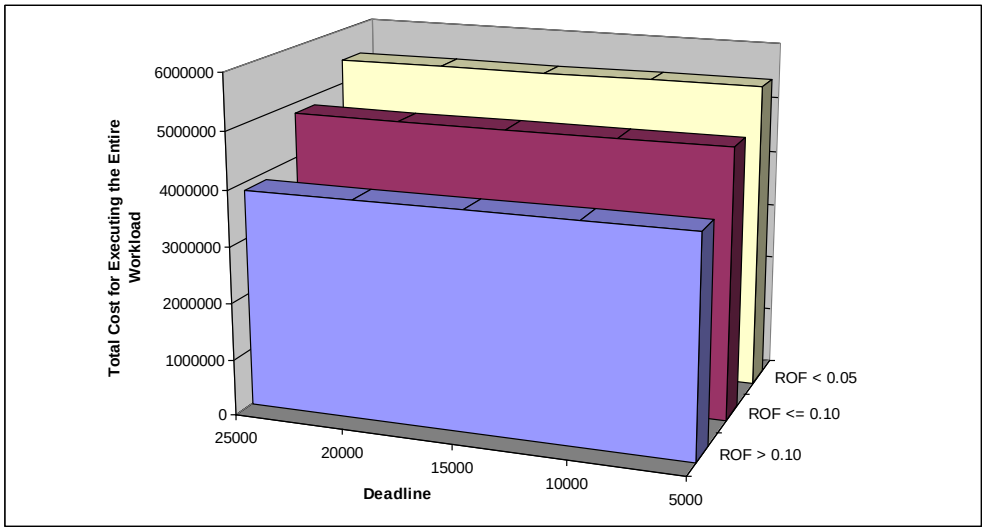


Figure 90: Total Execution Cost, Large Provider Running Large BoT Jobs.

For the BoT job owner, increasing the deadline and/or the ROF constraints decreases the cost of executing his/her BOT job, which is especially true when the ROF is increased; however, the decrease of the cost is limited and after a threshold the cost of executing the BoT job does not change, since the tasks are always assigned to the cheapest resource. Therefore, the BoT job owner does not gain any advantages by increasing the deadline and/or the ROF beyond the threshold.

6.6 Summary

In this chapter, the problem of scheduling Bag of Tasks application on Grid resources is modelled using Mixed Integer Programming (MIP) and an efficient algorithm, known as DRFC, is proposed and evaluated with simulation. Using simulation enabled the flexibility to investigate the DRFC algorithm under different scenarios.

The chapter began by looking at the Bag of Tasks application model, which dominated most of the Grid workload. This is followed by the limitations of current BoT scheduling algorithms. The use of the resources ROF is proposed in order to overcome most of the limitations. The chapter then described a model for minimizing the cost of executing BoT jobs whilst ensuring that the users' constraints are satisfied. This is followed by the formal mathematical model using the Mixed Integer Programming method. The MIP is NP-hard and finding an optimal solution in a reasonable amount of time is unfeasible; therefore, the DRFC algorithm is proposed to find a near-optimal solution in an acceptable time frame. The DRFC

algorithm is evaluated through simulation. The chapter provided a description of the simulation and presented the experiments design. This is followed by the resources model and the workload model, along with the exact parameters used. The chapter then presented the experiments and results. Notably, there were nine experiments to ensure that the DRFC performs as intended for a wide variety of problem instances.

The experiments were evaluated by two criteria: the number of jobs scheduled and the difference in the cost of executing a job when scheduled by DRFC and the optimal execution cost. The DRFC algorithm provided a near-optimal solution for all the experiments considered. Furthermore, the DRFC performance did not degraded with the use of large resource provider and large BoT jobs, which therefore makes the DRFC algorithm suitable for real-time scheduling. Finally, the effect of the user's constraints is analysed: by relaxing the constraints, the chance that the BoT job is scheduled increased, which is especially true if the resources available are limited or the BoT job submitted is large. Another observation is that, by relaxing the constraints, the cost of execution is reduced, which continues until, a threshold after that, the relaxation of the constraints does not provide any advantages to the user.

Chapter 7

Conclusion and Future Work

7.1 Summary of Work

The work presented in this thesis demonstrates a mathematical model to predict the risk of failure of a Grid resource using a discrete-time analytical model driven by reliability functions fitted to observed data. The model is also used to improve scheduling applications on the Grid.

- Chapter 2 introduces Grid computing as the broad area in which this research is conducted. Architectural philosophies—including OGSA and WSRF—are defined. Grid Middleware systems which enable access to heterogeneous resources are discussed. Furthermore, Service Level Agreements are presented as languages which formalise QoS requirements, and a discussion on a number of specifications actively used within the Grid research domain is presented. A discussion of Grid resource management identifies a number of limitations in Grid resources scheduling. In order to highlight these limitations, a number of scheduling algorithms are described. Finally, the prediction of application execution times and resources monitoring are also discussed as technologies required supporting scheduling in Grid environments.
- Chapter 3 introduces the definition of risk and its application in the real world. Methods for risk assessments—including qualitative and quantitative—are defined. A discussion of risks affecting Grid systems narrows the research to assessing the Grid resources risk of failures. In order to highlight risk assessment in Grid computing, a number of assessment methods applied in the field are described.
- Chapter 4 presents the motivation scenario for the Grid resources risk of failure model. The events causing resource failures are determined, and the method for measuring the risk of these events is presented. The need

for historical failure data is showcased, along with the data collection process. The statistical proprieties of the data—including the root cause of failures, the mean time to repair and time between failures—are analysed. Models to describe the time between failures and repair time are provided. Finally, the resource failures are tested against well-known probabilistic failure models in order to verify whether they can be used to model the Grid resources.

- Chapter 5 presents the mathematical model to predict the risk of failure of a Grid resource. The model selection and the modelling method are described. The model is then developed based on the different distributions of the failure data. The model is validated by comparing the model ROF with the observed ROF. Finally, the use of the model to rank resources and plan future investments is presented.
- Chapter 6 provides an overview of the Grid scheduling problem, and presents the Grid application model, as well as limitation of current scheduling algorithms. The use of resources ROF to overcome the current algorithms limitations and the Mixed Integer Programming model to minimise the cost of executing a BoT job whilst guaranteeing the user's requirements are presented. An algorithm to determine a near-optimal solution in an acceptable time frame is described. Moreover, the evaluation of the algorithm is carried out via simulation. The design of the experiments is showcased, along with the resource model and the workload model used in the experiments. The experiments compare the algorithm with the optimal scheduler, and were evaluated by two criteria: the number of jobs scheduled and difference in the cost of executing a job when scheduled by algorithm and the optimal execution cost.

7.2 Thesis Contribution

The aim of the work presented in this thesis is to increase the chances of Grid commercial take-up and to help building trust in the Grid. The main contributions of this thesis are summarised in the following points:

- The development of a mathematical model to predict the Grid resources risk of failure. A continuous time-varying Markov Model described the Grid resource availability. In order to solve the Markov model, there is the need to approximate the continuous-time process with discrete-time equivalents. The resulting discrete time-varying Markov Model is used to predict the resources risk of failure. The failure data collected from GOCDB are used to conduct the mathematical model and to compute the observed—or actual—ROF. The mathematical model is then evaluated by comparing the model-predicted ROF with the observed ROF. The two-sample t test—also known as Independent-Samples T Test—is used to compare the means of the two groups (observed and predicted risk of failure). The test showed that the difference between the predicted and observed risk of failure was statistically not significant. The mathematical model was developed after a detailed analysis of Grid resource failures using failure data collected from different Grid resources and spanning for three years. The analysis focused on the statistical properties of the failure data, including the root cause of failures, the mean time between failures, and the mean time to repair. The best model for the time between failures is the Weibull distribution, with decreasing hazard function rate. Repair times are much better modelled by a lognormal distribution than an exponential distribution.
- The development of an efficient algorithm—known as DRFC algorithm—was carried out in order to find a near-optimal execution cost for the cost minimising mathematical model. A greedy approach was considered to make the resulting resources allocation consistence and to ensure the resources were fully utilised. A simulation is used to evaluate the performance of the DRFC algorithm compared to the cost-minimising mathematical model optimal solution. There were two evaluation criteria: the first criterion is the percentage of the number of BoT jobs scheduled using the DRFC algorithm and the optimal schedule using Gurobi optimiser; whilst the second criterion is the average difference between the costs of executing a BoT job when scheduled with the DRFC algorithm, and the optimal costs of executing the BoT job when scheduled using Gurobi optimiser. The evaluation shows that finding the optimal allocation

of tasks is an intensive process requiring a long period of time and a powerful computer to compute with a huge internal memory. Thus, it is inefficient to use this method for real-time scheduling. The DRFC algorithm provided a good solution for a wide variety of resource providers and workloads. The difference in the execution cost between the optimal solution and the solution found using DRFC algorithm is minimal and, in most cases, was less than 1%. This therefore suggests that the DRFC algorithm is a superior choice for real-time scheduling of BoT jobs in Grid environments.

7.3 Future Work

There are many ways to further extend the work presented in this thesis. The most appealing ones are listed below:

- The risk assessment model presented in this work only considered the resources historical data. An extension to this model is to consider dynamic data, such as the current resource load or the availability of administrators to enhance the model, since the mean time to repair a resource is hugely influenced by the availability of administrators.
- Another extension to the risk model is to consider the internal components of a resource rather than considering a resource as a black-box. This extension model has different components failures, such as CPU, memory, hard drive, etc., and drives the resource risk of failure through campaigning all the components models.
- The risk assessment model did not consider the type and intensity of the workload running on a resource. However, there is evidence of a correlation between the type and intensity of the workload and the failure rate of the resource [170]. Importantly, extending the model to cater for this information will provide a more accurate prediction.
- The data used to develop the model were from research institutes; therefore, the mean time to repair all resources is very high as the sites do not have 24-hour support and there is no automatic monitoring which will report a resource failure when it occurs. It would be ideal to use data from

commercial Grid provider, if available, to further validate the risk assessment model.

- The risk assessment model was developed and evaluated analytically. Therefore, it would be beneficial to implement the model on a production Grid in order to evaluate the performance.
- The BoT scheduling algorithm did not take into account the time to stage input files and output files, the cost of the staging, or the reliability of the network. An extension to the algorithm could provide better cost-optimisation, such as executing the BoT job in a Grid system close to the input files in order to reduce the cost of data transfer.
- Other scheduling algorithms—such as minimising the BoT execution time without exceeding the user budget and ROF—would enable the Grid user to specify different constraints based on the job requirements.
- The scheduling algorithm was evaluated using simulation; accordingly, it would be ideal to implement the algorithm on a production Grid so as to evaluate the performance of the algorithm in a real environment.

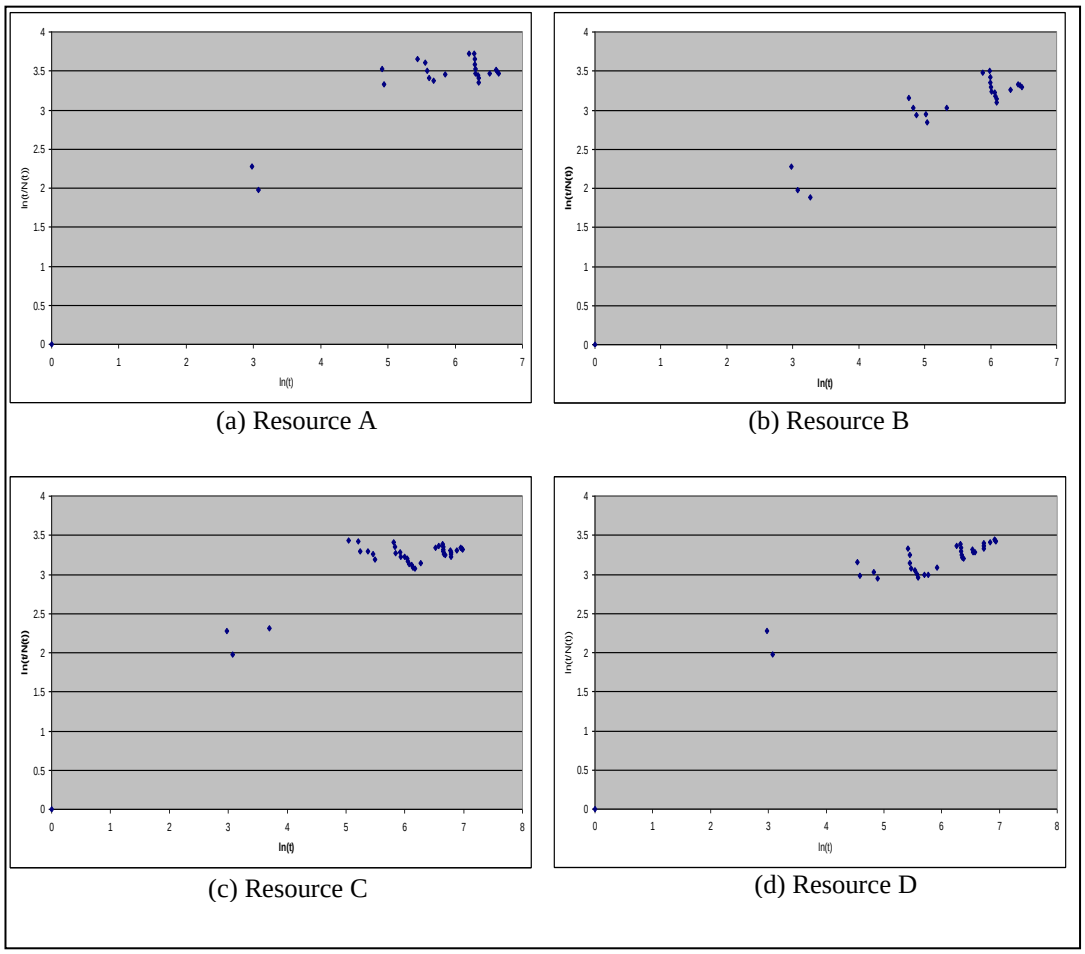
Appendix A

Table 16 shows a sample downtime record from GOCDB.

Table 16: A Sample Downtime Record.

Classification	*Severity*	*From*	*To*	*Description*
UNSCHEDULED	OUTAGE	17/02/2011 04:00	17/02/2011 17:00	Deploying kdump after kernel panics
UNSCHEDULED	OUTAGE	07-FEB-11 13:00:00	07-FEB-11 16:00:00	disk server crashed
SCHEDULED	AT_RISK	12-JAN-11 20:00:00	13-JAN-11 08:00:00	NREN network maintenance pssoible perturbation
UNSCHEDULED	OUTAGE	07-JAN-11 00:30:00	07-JAN-11 05:00:00	DNS failure
UNSCHEDULED	AT_RISK	01-JAN-11 01:00:00	03-JAN-11 13:00:00	CRAC failure
UNSCHEDULED	OUTAGE	15-DEC-10 19:45:00	16-DEC-10 08:40:00	Router down, top BDII unavailable
SCHEDULED	OUTAGE	03-DEC-10 05:00:00	03-DEC-10 07:00:00	Network maintenance
UNSCHEDULED	OUTAGE	02-OCT-10 17:53:00	02-OCT-10 21:25:00	CRAC failure
UNSCHEDULED	OUTAGE	17-SEP-10 13:30:00	22-SEP-10 15:45:00	Security stop
UNSCHEDULED	OUTAGE	26-AUG-10 22:00:00	27-AUG-10 16:05:00	CRAC failure
UNSCHEDULED	AT_RISK	19-AUG-10 10:00:00	19-AUG-10 10:30:00	Network maintenance
UNSCHEDULED	AT_RISK	19-JUL-10 09:14:00	26-JUL-10 12:30:00	Server room UPS batteries not charging
UNSCHEDULED	OUTAGE	16-JUL-10 09:14:00	20-JUL-10 09:14:00	Site down ! moving to a new comuter room.
UNSCHEDULED	OUTAGE	16-JUL-10 00:00:00	16-JUL-10 09:14:00	All hardware being relocated to new server room
SCHEDULED	OUTAGE	09-JUL-10 00:00:00	16-JUL-10 00:00:00	All hardware being relocated to new server room

Figures 91, 92, 93 & 94 show for the Duane plot and log-linear plot for the resources repair time. From the figures it is most likely that the Grid resources repair time cannot be modelled as NHPP.



(a) Resource A (b) Resource B

(c) Resource C (d) Resource D

Figure 91: The Duane Plot for Resources Repairs time, Site 1.

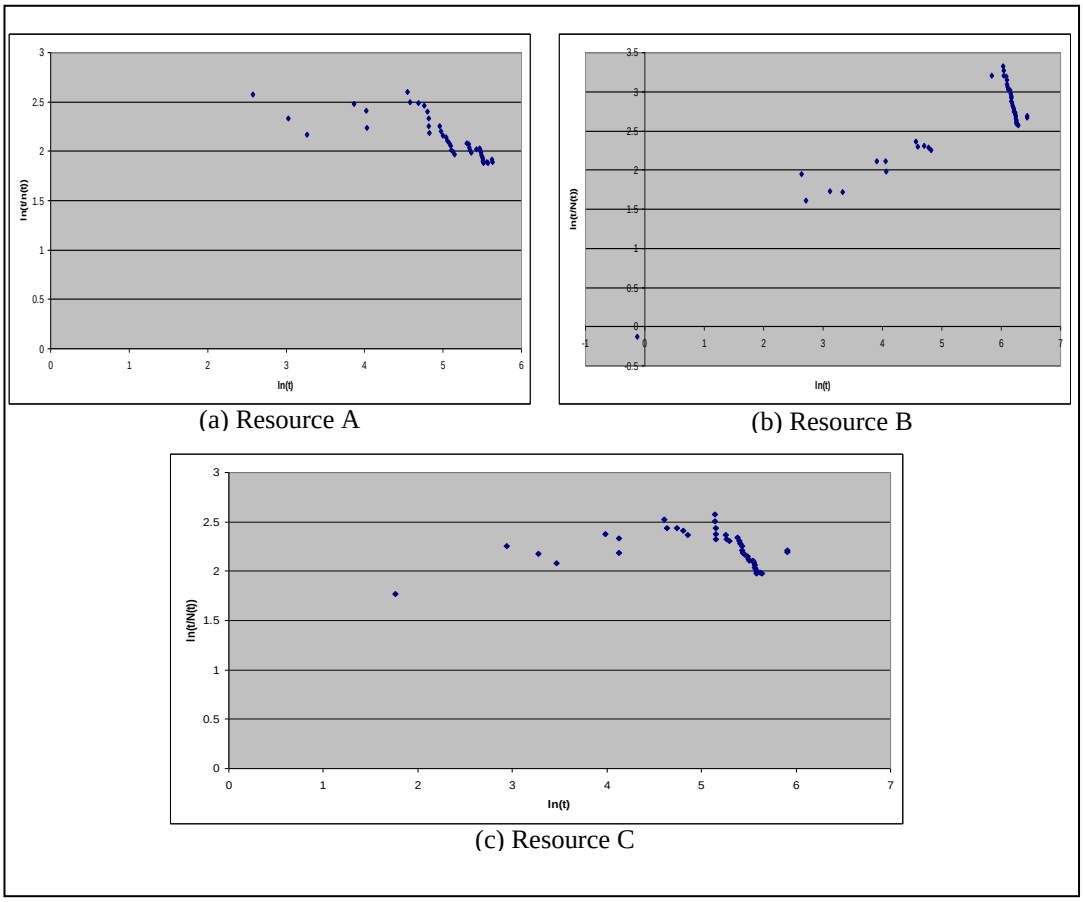


Figure 92: The Duane Plot for Resources Repairs time, Site 2.

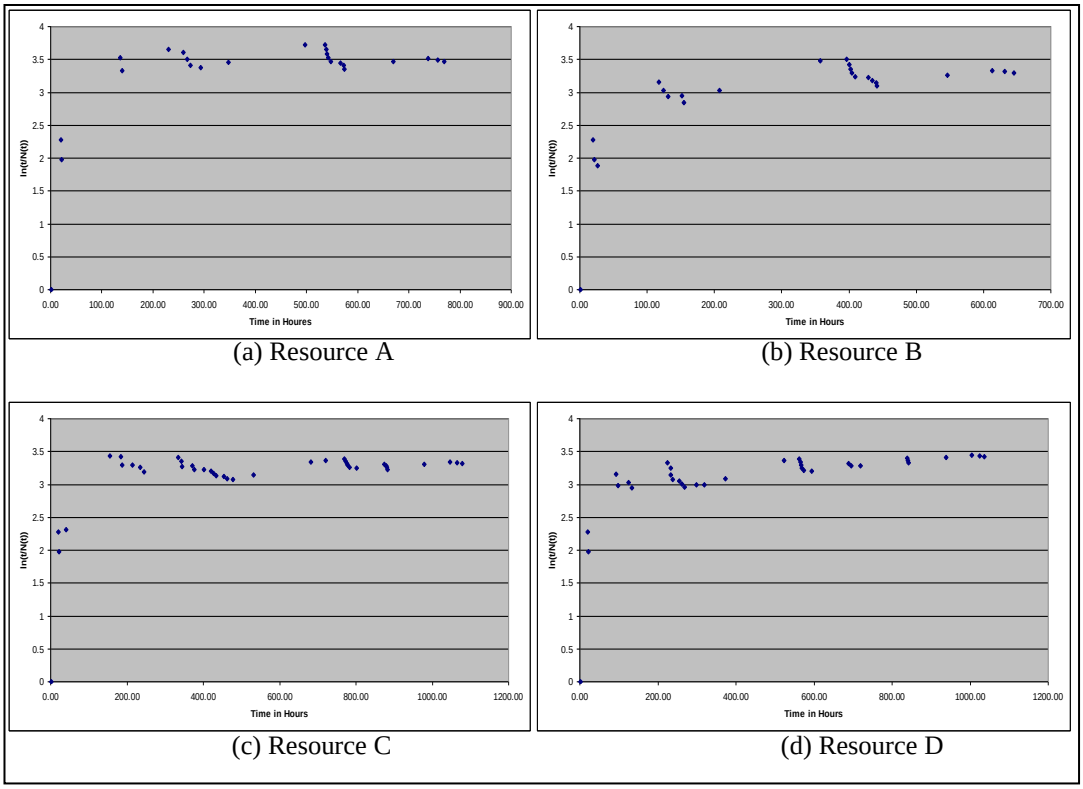
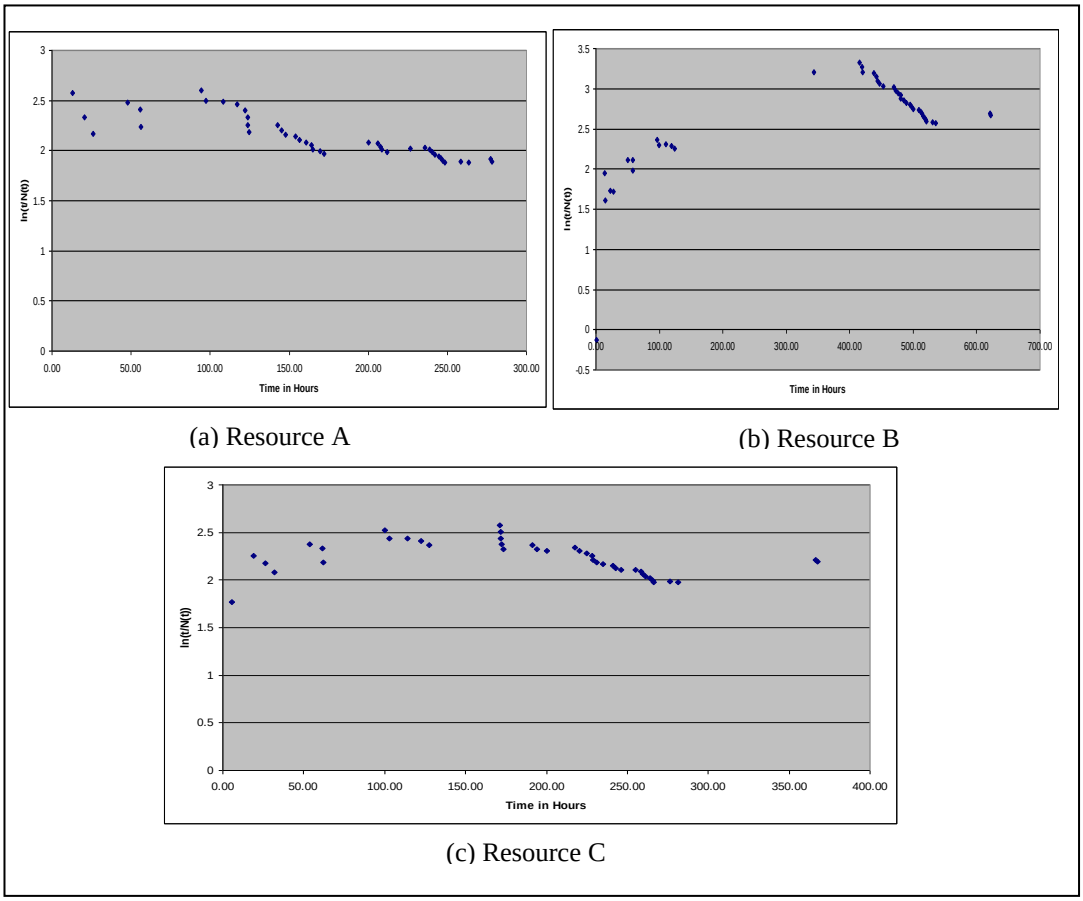


Figure 93: log-linear Paper for Resources Repairs time, Site 1.



(a) Resource A

(b) Resource B

(c) Resource C

Figure 94: log-linear Paper for Resources Repairs time, Site 2.

Appendix B

The following table are the results of the t-test.

Independent Samples Test For Observed and Predicted ROF Resource A Site 1.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	1.526	.218	1.397	258	.164	.01110	.00794	-.00455	.02674
Equal variances not assumed			1.397	254.904	.164	.01110	.00794	-.00455	.02674

Independent Samples Test For Observed and Predicted ROF Resource B Site 1.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	3.062	.081	.938	258	.349	.00566	.00603	-.00622	.01753
Equal variances not assumed			.938	254.222	.349	.00566	.00603	-.00622	.01753

Independent Samples Test For Observed and Predicted ROF Resource C Site 1.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	4.011	.046	1.683	258	.093	.01213	.00720	-.00206	.02631
Equal variances not assumed			1.683	251.506	.094	.01213	.00720	-.00206	.02631

Independent Samples Test For Observed and Predicted ROF Resource D Site 1.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17.894	.000	1.916	258	.056	.01545	.00806	-.00043	.03132
Equal variances not assumed			1.916	232.665	.057	.01545	.00806	-.00044	.03133

Independent Samples Test For Observed and Predicted ROF Resource A Site 2.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	11.505	.001	1.923	258	.056	.00607	.00316	-.00015	.01229
Equal variances not assumed			1.923	245.188	.056	.00607	.00316	-.00015	.01229

Independent Samples Test For Observed and Predicted ROF Resource B Site 2.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	18.329	.000	.875	258	.383	.00288	.00329	-.00360	.00936
Equal variances not assumed			.875	237.853	.383	.00288	.00329	-.00361	.00937

Independent Samples Test For Observed and Predicted ROF Resource C Site 2.

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	T	Df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	16.255	.000	1.697	258	.091	.00552	.00325	-.00088	.01192
Equal variances not assumed			1.697	240.073	.091	.00552	.00325	-.00089	.01193

Appendix C

The following experiments are carried out to ensure that the DRFC algorithm and the MIP solver have been implemented correctly in the simulations. These were carried out prior to the experiments described in chapter 6.

In all tests the values for the following parameters will be obtained manually and using simulation:

1. The assignment of tasks to resources.
2. Total execution cost.

For the DRFC algorithm; if the expected schedule and the simulation results agree, then this gives confidence that the algorithm is being applied correctly.

For the Gurobi optimizer; if the optimal schedule and the solver results agree, then this gives confidence that the MIP solver is implemented correctly. Furthermore, the MIP was also implemented using AMPL with CPLEX solver. The Gurobi optimizer results were compared with the AMPL results. Figure 95 illustrates a snippet of the AMPL code responsible for scheduling a single BoT job.

```
set Tasks;
set Resources;
  param P {Tasks} >  0;
  param I {Tasks} >  0;
param cost {Resources} > 0;
param speed {Resources} > 0;
param risk {Resources} > 0;
param A {Resources} >= 0;
param T = ; # The User Deadline
param R = ; # The User desire ROF
param time{Resources};
param x;
param y;
```

```
param total;
var In {Tasks,Resources} binary;
minimize Total_Value:
        sum {i in Tasks, j in Resources} (ceil(P[i]/speed[j])*cost[j]) * In[i,j];
subject to Weight_Limit { j in Resources}:
        sum {i in Tasks} ceil(P[i]/speed[j]) * In[i,j] <= T - A[j];
subject to Only_One {i in Tasks}:
        sum {j in Resources} In [i,j] = I[i];
subject to Risk  {i in Tasks, j in Resources}:
        risk[j] * In[i,j] <= R;
```

Figure 95: Snippet of the AMPL responsible for scheduling a single BoT job.

Test 1

Consider 4 identical resources and 3 BoT jobs. Tables 17 & 18 show the exact values used in the test.

Table 17: Resources Used for Test 1.

Number of Resources	*4*
MIPS	6000
Price per Time Unit	1
ROF	< 0.05

Table 18: BoT Jobs Used for Test 1.

BoT job	*1*	*2*	*3*
Arrival Time	0	50	100
Deadline	500	600	800
Number of Tasks	5	5	5
Desire ROF		< 0.05	
Execution time/task*		200	

* Execution time is based on a resource with 6000 MIPS

Test 2

Consider 4 resources and 3 BoT jobs. Tables 19 & 20 show the resources and the BoT jobs values.

Table 19: Resources Used for Test 2.

Resources	*1*	*2*	*3*	*4*
MIPS	6000	7000	8000	8000
Price per Time Unit	1	1	2	3
ROF	< 0.05	> 0.1	≤ 0.1	< 0.05

Table 20: BoT Jobs Used for Test 2.

BoT job	*1*	*2*	*3*
Arrival Time	0	200	400
Deadline	1000	1200	1600
Number of Tasks	5	5	5
Desire ROF	< 0.05	> 0.1	≤ 0.1
Execution time/task*	Between 200 and 600 in steps of 100		

* Execution time is based on a resource with 6000 MIPS

Test 3

Consider 5 resources and 4 BoT jobs. Tables 21 & 22 show the resources and the BoT jobs values.

Table 21: BoT Jobs Used for Test 3.

Resources	*1*	*2*	*3*	*4*	*5*
MIPS	6000	7000	7000	8000	8000
Price per Time Unit	1	2	3	4	5
ROF	≤ 0.1	> 0.1	< 0.05	≤ 0.1	< 0.05

Table 22: BoT Jobs Used for Test 3.

BoT job	*1*	*2*	*3*	*4*
Arrival Time	0	200	400	600
Deadline	1000	1200	1400	1600
Number of Tasks	5	5	5	5
Desire ROF	< 0.05	> 0.1	≤ 0.1	> 0.1
Execution time/task*	Between 200 and 600 in steps of 100			

* Execution time is based on a resource with 6000 MIPS

Result for Test 1

In this test the optimal schedule and the DRFC algorithm schedule are identical. Figure 96 shows the Gantt chart for the resulted schedule. The DRFC algorithm, the Gurobi optimizer, the AMPL solver and the manual solutions are all identical. The total execution cost for all the BoT jobs is 3000.

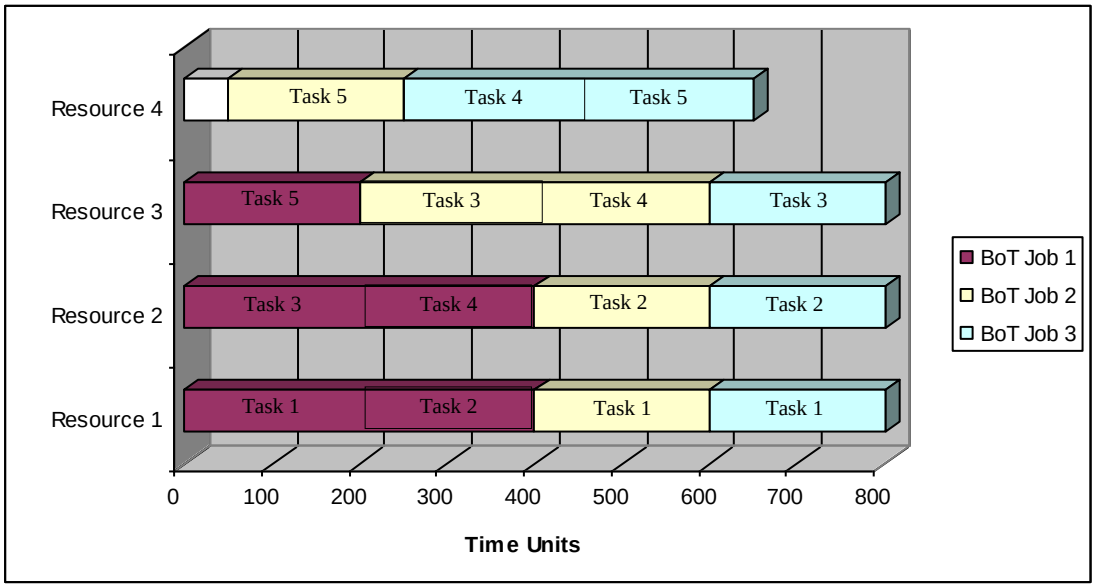


Figure 96: The Gantt chart for Test 1 Schedule.

Results Test 2

In this test also the optimal schedule and the DRFC schedule are identical. The total execution cost is 8544. Figure 97 shows the resulted Gantt chart scheduler.

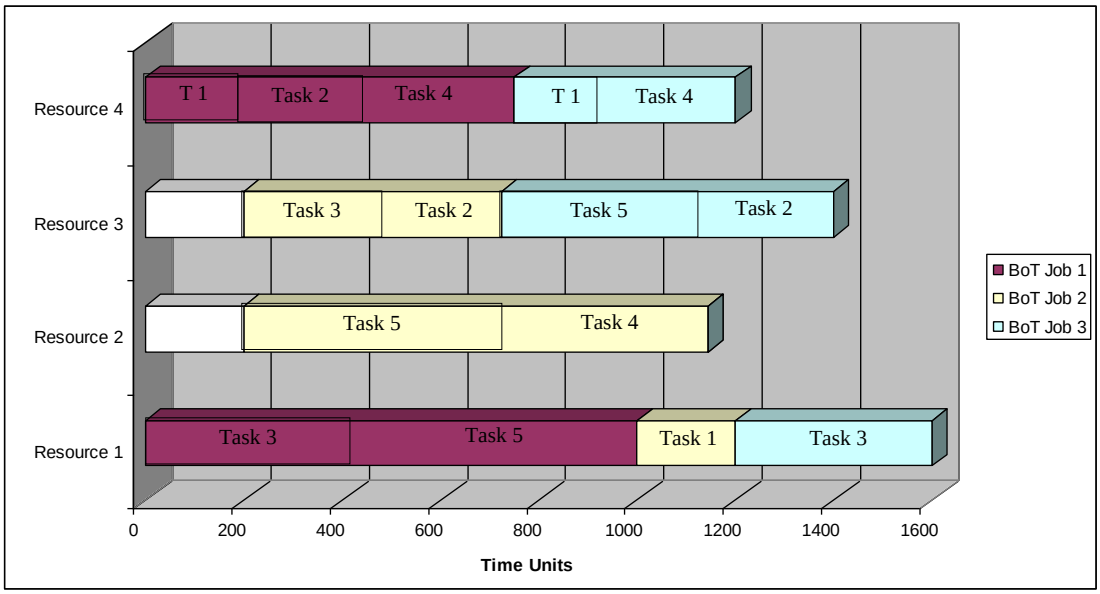


Figure 97: The Gantt chart for Test 2 Schedule.

Results Test 3

In this test also the optimal schedule and the DRFC schedule are identical. The total execution cost is 19934. Figure 98 shows the resulted Gantt chart scheduler.

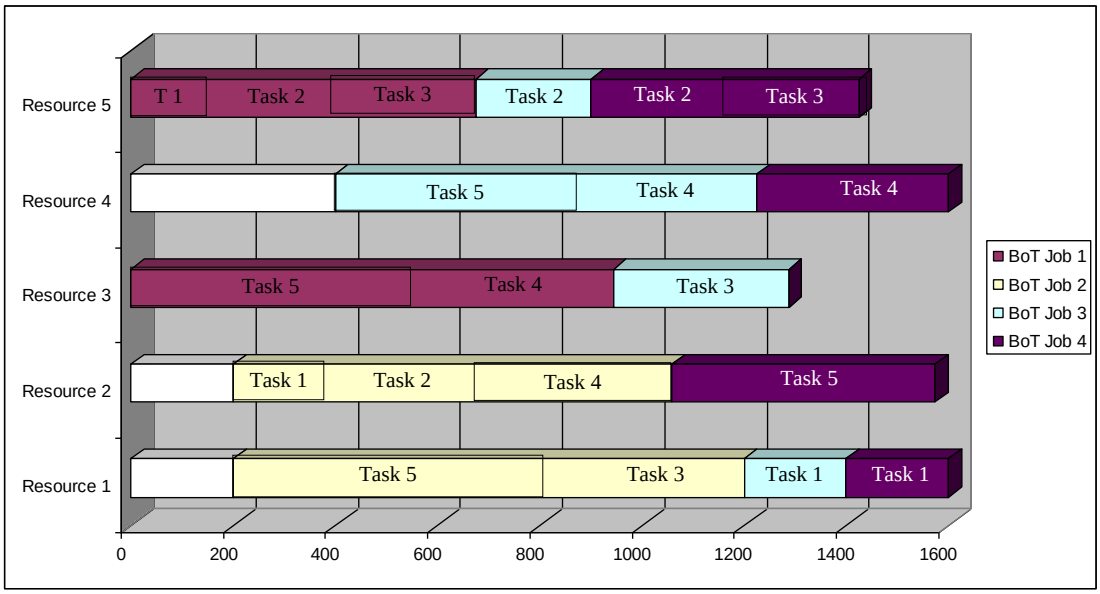


Figure 98: The Gantt chart for Test 3 Schedule.

References

1. Foster, I., *The Grid: A New Infrastructure for 21st Century Science*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 51-63.
2. Plaszczak, P. and R. Wellner, *Grid Computing : The Savvy Manager's Guide*. 2005, San Francisco: Elsevier/Morgan Kaufmann.
3. Foster, I. *Globus Toolkit Version 4: Software for Service-Oriented Systems*. in *FIP International Conference on Network and Parallel Computing*. 2005: Springer-Verlag LNCS 3779.
4. Misev, A. and E. Atanassov, *User Level Grid Quality of Service*, in *Large-Scale Scientific Computing*, I. Lirkov, S. Margenov, and J. Wasniewski, Editors. 2010, Springer Berlin / Heidelberg. p. 507-514.
5. Al-Ali, R., et al. *QoS Support for High-Performance Scientific Grid Applications*. in *IEEE International Symposium on Cluster Computing and the Grid, CCGrid*. 2004.
6. Lee, H., et al., *A Resource Management and Fault Tolerance Services in Grid Computing*. Journal of Parallel and Distributed Computing, 2005. **65**(11): p. 1305-1317.
7. *Oxford English Dictionary*. Available from: http://dictionary.oed.com.
8. Koller, G.R., *Risk Assessment and Decision Making in Business and Industry: a Practical Guide*. 2nd ed. 2005, Boca Raton, FL: Chapman & Hall/CRC.
9. Modarres, M., *Risk Analysis In Engineering: Techniques, Tools, and Trends*. 2006, Boca Raton: Taylor & Francis.
10. Blischke, W.R. and D.N.P. Murthy, *Reliability: Modeling, Prediction, and Optimization*. Wiley Series In Probability And Statistics. 2000, New York: Wiley.
11. Abbas, A., *Grid computing: A Practical Guide to Technology and Applications*. 2004, Hingham, Mass: Charles River Media.
12. Roure, D.D., et al., *The Evolution of the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley: New York. p. 65-100.
13. Foster, I., et al., *Software Infrastructure for the I-WAY High-Performance Distributed Computing Experiment*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 101-115.
14. Foster, I. and C. Kesselman, *The Grid in a Nutshell*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 3-13.

15. Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 169-197.

16. Foster, I. and C. Kesselman, *Concepts and Architecture*, in *The grid : blueprint for a new computing infrastructure*, I. Foster and C. Kesselman, Editors. 2004, Morgan Kaufmann: Amsterdam ; Boston. p. 37-64.

17. *The grid : blueprint for a new computing infrastructure*, ed. I. Foster and C. Kesselman. 1999, San Francisco: Morgan Kaufmann Publishers.

18. *The grid : blueprint for a new computing infrastructure*. 2nd ed, ed. I. Foster and C. Kesselman. 2004, Amsterdam ; Boston: Morgan Kaufmann.

19. Krauter, K., R. Buyya, and M. Maheswaran, *A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing.* Software: Practice and Experience, 2002. **32**(2): p. 135-164.

20. Foster, I. and C. Kesselman, *Chapter 2: Computational Grid*, in *The grid : blueprint for a new computing infrastructure*. 1999, Morgan Kaufmann Publishers: San Francisco. p. 15-53.

21. Chervenak, A., et al., *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets* Journal of Network and Computer Applications, July 2000. **23**(3): p. 187-200.

22. Foster, I., et al., *The Physiology of the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 217-249.

23. *Enabling Grids for E-sciencE.* Available from: http://www.eu-egee.org/.

24. *Grid 5000*. Available from: https://www.grid5000.fr.

25. *National Grid Service.* Available from: http://www.grid-support.ac.uk/.

26. *European Grid Infrastructure EGI.eu*. Available from: http://www.egi.eu/.

27. *Integrated Sustainable Pan-European Infrastructure for Researchers in Europe*. Available from: http://www.egi.eu/projects/egi-inspire/.

28. *A worldwide e-infrastructure for computational neuroscientists*. Available from: http://www.outgrid.eu.

29. *A Grid-Based e-Infrastructure for data archiving/ communication and computationally intensive applications in the medical sciences.* Available from: http://www.neugrid.eu.

30. *CBRAIN Project*. Available from: http://www.cbrain.mcgill.ca.

31. *LONI - Laboratory of Neuro Imaging at UCLA*. Available from: http://www.loni.ucla.edu/.

32. *Desktop Grids for International Scientific Collaboration*. Available from: http://degisco.eu/.

33. *EDGeS: Enabling Desktop Grids for e-Science.* Available from: http://www.edges-grid.eu/.

34. Austin, J., et al., *Predictive Maintenance: Distributed Aircraft Engine Diagnostics*, in *The grid : blueprint for a new computing infrastructure*, I.

Foster and C. Kesselman, Editors. 2004, Morgan Kaufmann: Amsterdam ; Boston. p. 69-79.

35. Djemame, K. and M. Haji. *Grid Application Performance Prediction: a Case Study in BROADEN*. in *In Proceedings of the 1st International Workshop On Verification and Evaluation of Computer and Communication Systems (VECoS'2007)*. 2007. Algiers, Algeria.
36. *The AssessGrid Project*. Available from: http://www.assessgrid.eu.
37. Djemame, K., et al., *Introducing Risk Management into the Grid*, in *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. 2006, IEEE Computer Society.
38. Ioannis, K., et al., *IBHIS: Integration Broker for Heterogeneous Information Sources*, in *Proceedings of the 27th Annual International Conference on Computer Software and Applications*. 2003, IEEE Computer Society: Dallas, Texas, USA.
39. Townend, P., et al., *The e-Demand project: A Summary* in *4th UK e-Science All-hands Conference AHM 2005*. 2005: Nottingham, UK.
40. Berman, F., G. Fox, and T. Hey, *The Grid: Past, Present, Future*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 9-50.
41. *Open Grid Forum (OGF)*. Available from: http://www.ogf.org.
42. Foster, I., C. Kesselman, and S. Tuecke, *The Open Grid Services Architecture*, in *The grid : blueprint for a new computing infrastructure*, I. Foster and C. Kesselman, Editors. 2004, Morgan Kaufmann: Amsterdam ; Boston. p. 215-258.
43. *Web Services Architecture*. World Wide Web Consortium:[Available from: http://www.w3.org/TR/ws-arch/.
44. *World Wide Web Consortium. SOAP Version 1.2*. Available from: http://www.w3.org/TR/soap/.
45. Christensen, E., et al. *Web Services Description Language (WSDL) 1.1*. 2001 Available from: http://www.w3.org/TR/wsdl
46. Brittenham, P. *An overview of the Web Services Inspection Language*. 1 Jun 2002 Available from: https://www.ibm.com/developerworks/webservices/library/ws-wsilover/.
47. Tuecke, S., et al. *Open Grid Services Infrastructure (OGSI) Version 1.0*. Available from: http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf.
48. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available from: http://www.w3.org/TR/REC-xml/.
49. Czajkowski, K., et al. *From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution*. 2004 Available from: http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf.
50. Czajkowski, K., et al. *The WS-Resource Framework*. 2004 Available from: http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

51. Baker, M., R. Buyya, and D. Laforenza, *Grids and Grid Technologies for Wide-Area Distributed Computing.* Software: Practice and Experience, 2002. **32**: p. 1437 - 1466.

52. *The Globus Project*. Available from: http://www.globus.org/.

53. Czajkowski, K., et al. *A Resource Management Architecture for Metacomputing Systems*. in *Proceedings of Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'98 Workshop* 1998. Orlando, Florida, USA.

54. Allcock, W., et al. *The Globus Striped GridFTP Framework and Server*. in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing* 2005. Los Alamitos, CA, USA: IEEE Computer Society.

55. Madduri, R.K., C.S. Hood, and W.E. Allcock. *Reliable File Transfer in Grid Environments*. in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*. 2002.

56. Konstantinos, K., et al., *Introduction to OGSA-DAI Services*. Lecture Notes in Computer Science : Scientific Applications of Grid Computing. 2005.

57. *UNICORE*. Available from: http://www.unicore.eu/.

58. Sakellariou, R., et al., *UNICORE: A Grid Computing Environment*, in *Euro-Par 2001 Parallel Processing*. 2001, Springer Berlin / Heidelberg. p. 825-834.

59. Laure, E., et al., *Middleware for the Next Generation Grid Infrastructure*, in *Computing in High Energy and Nuclear Physics (CHEP)*. 2004: Interlaken, Switzerland.

60. Laure, E., et al., *Programming the Grid Using gLite.* Computational Methods in Science and Technology 2006. **12**(1): p. 33-45.

61. Wu, Y., et al., *Grid Middleware in China.* International Journal of Web and Grid Services 2007. **3**(4): p. 371 - 402.

62. *OMII-UK*. Available from: http://www.omii.ac.uk.

63. Sahai, A., et al. *Specifying and Monitoring Guarantees in Commercial Grids through SLA*. in *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*. 2003.

64. Czajkowski, K., et al., *Grid Service Level Agreements: Grid resource management with intermediaries* in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 119-134.

65. Czajkowski, K., et al., *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*. Lecture Notes in Computer Science. 2002. 153-183.

66. Leff, A., J.T. Rayfield, and D.M. Dias, *Service-Level Agreements and Commercial Grids.* Internet Computing, IEEE, 2003. **7**(4): p. 44-50.

67. Ludwig, H., et al., *A Service Level Agreement Language for Dynamic Electronic Services.* Electronic Commerce Research, 2003. **3**(1): p. 43-59.

68. Keller, A. and H. Ludwig, *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services.* Journal of Network and Systems Management, 2003. **11**(1): p. 57-81.

69. *Grid Resource Allocation and Agreement Protocol Working Group (GRAAP-WG)*. Available from: http://forge.ggf.org/sf/projects/graap-wg.

70. Andrieux, A., et al., *Web Services Agreement Specification (WS-Agreement)*. 2007, Open Grid Forum.

71. Krämer, B., et al., *Improving Temporal-Awareness of WS-Agreement*, in *Service-Oriented Computing – ICSOC 2007*, Springer Berlin / Heidelberg. p. 193-206.

72. Heiko, L., D. Asit, and K. Robert, *Cremona: an Architecture and Library for Creation and Monitoring of WS-Agreents*, in *Proceedings of the 2nd international conference on Service oriented computing*. 2004, ACM: New York, NY, USA.

73. Lumb, I. and C. Smith, *Scheduling Attributes and Platform LSF*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 171-182.

74. Nitzberg, B., J.M. Schopf, and J.P. Jones, *PBS PRO: Grid Computing and Scheduling Attributes*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 183-190.

75. Kannan, S., et al., *Workload Management with LoadLeveler*. IBM RedBookes. 2001.

76. Grimshaw, A.S., et al., *From Legion to Avaki: the Persistence of Vision*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 265-298.

77. Natrajan, A., M.A. Humphrey, and A.S. Grimshaw, *Grid Resourcs Management in Legion*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 145-160.

78. Roy, A. and L. Miron, *Condor and Preemptive Resume Scheduling*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 135-144.

79. Thain, D., T. Tannenbaum, and M. Livny, *Condor and the Grid*, in *Grid Computing : Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, Editors. 2003, J. Wiley,: New York. p. 299-335.

80. Schopf, J.M., *Ten Action When Grid Scheduling*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 15-23.

81. Schwiegelshohn, U. and R. Yahyapour, *Attributes for Communication Between Grid Scheduling Instances*, in *Grid resource management : state of the art*

and future trends, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 41-52.

82. Czajkowski, K., et al. *Grid Information Services for Distributed Resource Sharing*. in *Proceedings 10th IEEE International Symposium on High Performance Distributed Computing*. 2001.
83. *Information Services (MDS) : Key Concepts*. Available from: http://www.globus.org/toolkit/docs/4.0/info/key-index.html.
84. Iamnitchi, A. and I. Foster, *On Fully Decentralized Resource Discovery in Grid Environments*, in *Grid Computing — GRID 2001*. 2001, Springer Berlin / Heidelberg. p. 51-62.
85. Cai, M., et al. *MAAN: a Multi-Attribute Addressable Network for Grid Information Services*. in *Proceedings Fourth International Workshop on Grid Computing*. 2003.
86. Xhafa, F. and A. Abraham, *Computational Models and Heuristic Methods for Grid Scheduling Problems.* Future Gener. Comput. Syst., 2010. **26**(4): p. 608-621.
87. Iosup, A., et al., *The Characteristics and Performance of Groups of Jobs in Grids*, in *Euro-Par 2007 Parallel Processing*. 2007, Springer Berlin / Heidelberg. p. 382-393.
88. Jia, Y. and B. Rajkumar, *A Taxonomy of Scientific Workflow Systems for Grid Computing.* SIGMOD Rec., 2005. **34**(3): p. 44-49.
89. *Oracle Grid Engine*. Available from: http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.
90. Keller, A. and A. Reinefeld. *CCS Resource Management in Networked HPC Systems*. in *Proceedings of Heterogeneous Computing Workshop*. 1998. Orlando, FL , USA
91. Jackson, D.B., *Grid Scheduling with Maui/Silver*, in *Grid resource management : state of the art and future trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2004, Kluwer Academic Publishers: Boston. p. 161-170.
92. Buyya, R., D. Abramson, and S. Venugopal, *The Grid Economy.* Proceedings of the IEEE, 2005. **93**(3): p. 698.
93. Nudd, G.R., et al., *Pace--A Toolset for the Performance Prediction of Parallel and Distributed Systems.* International Journal of High Performance Computing Applications, 2000. **14**(3): p. 228-251.
94. Kapadia, N.H., J.A.B. Fortes, and C.E. Brodley, *Predictive Application-Performance Modeling in a Computational Grid Environment*, in *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*. 1999, IEEE Computer Society: Redondo Beach, California, USA.
95. Dushay, N., J. French, and C. Lagoze, *Predicting Indexer Performance in a Distributed Digital Library*, in *Research and Advanced Technology for Digital Libraries*. 1999, Springer Berlin / Heidelberg. p. 852-852.

96. Brandolese, C., et al., *Source-level Execution Time Estimation of C Programs*, in *Proceedings of the ninth international symposium on Hardware/software codesign*. 2001, ACM: Copenhagen, Denmark.

97. Krishnaswamy, S., S.W. Loke, and A. Zaslavsky, *Estimating Computation Times of Data-Intensive Applications.* IEEE Distributed Systems Online, 2004. **5**(4).

98. Matsunaga, A. and J. Fortes. *On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications*. in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. 2010. Melbourne, Australia: IEEE Computer Society.

99. Minh, T.N. and L. Wolters. *Using Historical Data to Predict Application Runtimes on Backfilling Parallel Systems*. in *18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 2010. Pisa, Italy.

100. Nadeem, F., M.M. Yousaf, and M. Ali. *Grid Performance Prediction: Requirements, Framework, and Models*. in *International Conference on Emerging Technologies, 2006. ICET*. 2006. Peshawar, Pakistan.

101. Faerman, M., et al. *Adaptive Performance Prediction for Distributed Data-Intensive Applications*. in *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*. 1999. Portland, Oregon, United States: ACM.

102. Maheswaran, M., et al., *Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems*, in *Proceedings of the Eighth Heterogeneous Computing Workshop*. 1999, IEEE Computer Society: San Juan , Puerto Rico

103. Casanova, H., et al. *Heuristics for Scheduling Parameter Sweep Applications in Grid Environments*. in *Proceedings of the 9th Heterogeneous Computing Workshop*. 2000. Cancun , Mexico IEEE Computer Society.

104. Oscar, H.I. and E.K. Chul, *Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors.* Journal of the ACM, 1977. **24**(2): p. 280-289.

105. Berman, F., et al., *Adaptive Computing on the Grid Using AppLeS.* IEEE Transactions on Parallel and Distributed Systems, 2003. **14**(4): p. 369.

106. Francine, D.B., et al., *Application-Level Scheduling on Distributed Heterogeneous Networks,* in *Proceedings of the 1996 ACM/IEEE conference on Supercomputing*. 1996, IEEE Computer Society: Pittsburgh, Pennsylvania, United States.

107. Cirne, W., et al. *Running Bag-of-Tasks Applications on Computational Grids: the MyGrid Approach*. in *Proceedings of the International Conference on Parallel Processing*. 2003. Kaohsiung, Taiwan.

108. Lee, Y.C. and A.Y. Zomaya, *A Grid Scheduling Algorithm for Bag-of-Tasks Applications Using Multiple Queues with Duplication*, in *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse*. 2006, IEEE Computer Society: Honolulu, HI USA.

109. Lee, Y.C. and A.Y. Zomaya, *Practical Scheduling of Bag-of-Tasks Applications on Grids with Dynamic Resilience.* IEEE Trans. Comput., 2007. **56**(6): p. 815-825.

110. *OurGrid*. Available from: http://www.ourgrid.org/.

111. Marco, A.S.N., et al., *Transparent Resource Allocation to Exploit Idle Cluster Nodes in Computational Grids*, in *Proceedings of the First International Conference on e-Science and Grid Computing*. 2005, IEEE Computer Society: Melbourne, Australia.

112. De Rose, C.A.F., et al., *Allocation Strategies for Utilization of Space-Shared Resources in Bag of Tasks Grids.* Future Generation Computer Systems, 2008. **24**(5): p. 331-341.

113. Buyya, R., D. Abramson, and J. Giddy. *An Evaluation of Economy-Based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications* in *the 2nd International Workshop on Active Middleware Services (AMS 2000)*. 2000. Pittsburgh, USA: Kluwer Press.

114. Buyya, R., D. Abramson, and J. Giddy. *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*. in *Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region* 2000. Beijing, China.

115. Abramson, D., R. Buyya, and J. Giddy, *A Computational Economy for Grid Computing and Its Implementation in the Nimrod-G Resource Broker*. Future Generation Computer Systems, 2002. **18**(8): p. 1061-1074.

116. Buyya, R., et al., *Economic Models for Resource Management and Scheduling in Grid Computing.* Concurrency and Computation: Practice and Experience, 2002. **14**(13-15): p. 1507-1542.

117. Buyya, R., et al., *Scheduling Parameter Sweep Applications on Global Grids: a Deadline and Budget Constrained Cost-Time Optimization Algorithm.* Software—Practice & Experience, 2005. **35**(5): p. 491-512.

118. Buyya, R. and M. Murshed, *GridSim: a Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing.* The Journal of Concurrency and Computation: Practice and Experience, 2002. **14**(13-15): p. 1175-1220.

119. Kumar, S., K. Dutta, and V. Mookerjee, *Maximizing Business Value by Optimal Assignment of Jobs to Resources in Grid Computing* European Journal of Operational Research, 2009. **194**(3): p. 856 - 872

120. Macías, M., et al., *Enforcing Service Level Agreements Using an Economically Enhanced Resource Manager*, in *Economic Models and Algorithms for Distributed Systems*, D. Neumann, et al., Editors. 2010, Birkhäuser: Basel. p. 109-127.

121. *SORMA - Self-Organizing ICT Resource Management*. Available from: http://www.im.uni-karlsruhe.de/sorma/.

122. Menascé, D.A. and E. Casalicchio, *QoS in Grid Computing.* IEEE Internet Computing, 2004. **8**(4): p. 85-87.

123. Kurowski, K., A. Oleksiak, and J. Weglarz, *Multicriteria, Multi-User Scheduling in Grids with Advance Reservation.* Journal of Scheduling 2010. **13**(5): p. 493-508.

124. Tierney, B., et al. *A Grid Monitoring Architecture*. 2002 Available from: http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-3.pdf.

125. Rich, W., T.S. Neil, and H. Jim, *The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing.* Future Gener. Comput. Syst., 1999. **15**(5-6): p. 757-768.

126. Brian, T., et al., *The NetLogger Methodology for High Performance Distributed Systems Performance Analysis*, in *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*. 1998, IEEE Computer Society.

127. Dan, G., et al., *Dynamic Monitoring of High-Performance Distributed Applications*, in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. 2002, IEEE Computer Society.

128. Massie, M.L., B.N. Chun, and D.E. Culler, *The Ganglia Distributed Monitoring System: Design, Implementation, and Experience.* Parallel Computing, 2004. **30**(7): p. 817-840.

129. Ribler, R.L., et al. *Autopilot: Adaptive Control of Distributed Applications*. in *Proceedings. The Seventh International Symposium on High Performance Distributed Computing*. 1998. Chicago, IL , USA: IEEE Computer Society Press.

130. Randy, L.R., S. Huseyin, and A.R. Daniel, *The Autopilot Performance-Directed Adaptive Control System.* Future Generation Computer Systems, 2001. **18**(1): p. 175-187.

131. Serafeim, Z. and S. Rizos, *A Taxonomy of Grid Monitoring Systems.* Future Gener. Comput. Syst., 2005. **21**(1): p. 163-188.

132. Hubbard, D.W., *The Failure Of Risk Management : Why It's Broken And How To Fix It*. 2009, Hoboken, N.J.: Wiley.

133. Aven, T., *Risk Analysis: Assessing Uncertainties Beyond Expected Values and Probabilities*. 2008, Chichester, England ; Hoboken, NJ: Wiley.

134. Bartlett, J., et al., *Project Risk Analysis and Management Guide*. 2nd ed. 2004, High Wycombe: Association for Project Management.

135. Simon, P., D. Hillson, and K. Newland, *PRAM: Project Risk Analysis and Management Guide*. 1997, Norwich: Association for Project Management.

136. Bennett, J.C., et al., *Risk Analysis Techniques and Their Application to Software Development.* European Journal of Operational Research, 1996. **95**(3): p. 467-475.

137. White, D., *Application of Systems Thinking to Risk Management: a Review of the Literature.* Management Decision, 1995. **33**(10): p. 35-45.

138. Merna, T. and F.F. Al-Thani, *Corporate Risk Management*. 2nd ed. 2008, Chichester, England ; Hoboken, NJ: Wiley.

139. *Security Research*. Available from: http://www.securityresearch.at/en/is-services/risk-management/.

140. Carr, M., et al., *Taxonomy-Based Risk Identification*. SEI Technical Report SEI-93-TR-006, Pittsburgh, Pennsylvania: Software Engineering Institute, 1993.

141. Freeman, J.W., T.C. Darr, and R.B. Neely. *Risk Assessment for Large Heterogeneous Systems*. in *Computer Security Applications Conference*. 1997.

142. Haimes, Y.Y., *Total Risk Management.* Risk Analysis, 1991. **11**(2): p. 169-171.

143. Neumann, P.G., *Some Computer-Related Disasters and Other Egregious Horrors.* Aerospace and Electronic Systems Magazine, IEEE, 1986. **1**(10): p. 18-19.

144. Ewusi-Mensah, K., *Software Development Failures: Anatomy of Abandoned Projects*. 2003, Cambridge, MA: MIT Press.

145. Glass, R.L., *Software runaways*. 1998, Upper Saddle River, N.J.: Prentice Hall PTR ; London : Prentice-Hall International (UK).

146. Boehm, B.W., *Software Risk Management: Principles and Practices.* IEEE Software, 1991. **8**(1): p. 32-41.

147. Schmidt, R., et al., *Identifying Software Project Risks: An International Delphi Study.* Journal of Management Information Systems, 2001. **17**(4): p. 5-36.

148. Barki, H., S. Rivard, and J. Talbot, *Toward an Assessment of Software Development Risk.* Journal of Management Information Systems, 1993. **10**(2): p. 203-225.

149. Moynihan, T., *How Experienced Project Managers Assess Risk.* IEEE Software, 1997. **14**(3): p. 35-41.

150. Wallace, L. and M. Keil, *Software Project Risks and their Effect on Outcomes.* Communications of the ACM - Human-computer etiquette, 2004. **47**(4): p. 68-73.

151. Keil, M., et al., *A Framework for Identifying Software Project Risks.* Communications of the ACM, 1998. **41**(11): p. 76-83.

152. Verdon, D. and G. McGraw, *Risk Analysis in Software Design.* IEEE Security & Privacy, 2004. **2**(4): p. 79-84.

153. *The European Network and Information Security Agency (ENISA)*. Available from: http://www.enisa.europa.eu/.

154. Armbrust, M., et al., *A view of cloud computing.* Communications of the ACM April 2010 **53**(4): p. 50-58.

155. *Cloud Computing Benefits, risks and recommendations for information security*. November, 2009 Available from: http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport.

156. *The context-aware data-centric information sharing (Consequence)*. Available from: http://www.consequence-project.eu.

157. *SLA@SOI*. Available from: http://sla-at-soi.eu/.

158. *D.A6a Predictable / Manageable Service Engineering Methodology and Prediction Services*. September 2010 Available from: http://sla-at-soi.eu/wp-content/uploads/2009/07/D.A6a-M26-PredictableServiceEngineeringMethodology.pdf.

159. *Risk Management Evaluation*. 31/10/2006 Available from: http://www.assessgrid.eu/fileadmin/AssessGrid/usermounts/publications/deliverables/AssessGrid_D.1.2_Risk_Management_Evaluation.pdf.

160. Boehm, B. *Software Risk Management*. in *Proceedings of the 2nd European Software Engineering Conference*. 1989. Coventry, UK: Springer Berlin / Heidelberg.

161. *ISO/IEC 27005:2008 Information technology -- Security techniques -- Information security risk management*. 2008 Available from: http://www.iso.org/iso/catalogue_detail?csnumber=42107.

162. *D 3.1 Consultant Service and Dynamic Risk Assessment*. March 2008 Available from: http://www.assessgrid.eu/fileadmin/AssessGrid/usermounts/publications/deliverables/ASSESSGRID_D3.1_DYNAMIC_RISK.pdf.

163. Plank, J.S. and W.R. Elwasif. *Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems*. in *Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, Digest of Papers*. 1998. Munich , Germany

164. Nitin, H.V., *A Case for Two-Level Distributed Recovery Schemes*. ACM SIGMETRICS Performance Evaluation Review, 1995. **23**(1): p. 64-73.

165. Kavanaugh, G.P. and W.H. Sanders. *Performance Analysis of Two Time-Based Coordinated Checkpointing Protocols*. in *Proceedings Pacific Rim International Symposium on Fault-Tolerant Systems*. 1997. Taipei , Taiwan.

166. Long, D., A. Muir, and R. Golding. *A Longitudinal Survey of Internet Host Reliability*. in *Proceedings 14th Symposium on Reliable Distributed Systems*. 1995. Bad Neuenahr , Germany.

167. Crow, L.H. and N.D. Singpurwalla, *An Empirically Developed Fourier Series Model for Describing Software Failures*. IEEE Transactions on Reliability, 1984. **R-33**(2): p. 176-183.

168. Zadeh, L.A., *Fuzzy sets as a basis for a theory of possibility*. Fuzzy Sets and Systems, 1999. **100**(Supplement 1): p. 9-34.

169. *D 4.1 Advanced Risk Assessment*. September 2008 Available from: http://www.assessgrid.eu/fileadmin/AssessGrid/usermounts/publications/deliverables/ASSESSGRID_D4.1_Advanced_Risk_Assessment.pdf.

170. Schroeder, B. and G.A. Gibson. *A large-scale study of failures in high-performance computing systems*. in *International Conference on Dependable Systems and Networks, 2006. DSN 2006.* 2006.

171. Lin, T.T.Y. and D.P. Siewiorek, *Error log analysis: statistical modeling and heuristic trend analysis.* Transactions on Reliability, IEEE, 1990. **39**(4): p. 419.

172. Taliver, H., P.M. Richard, and D.N. Thu, *Improving cluster availability using workstation validation.* SIGMETRICS Perform. Eval. Rev., 2002. **30**(1): p. 217-227.

173. Jun, X., Z. Kalbarczyk, and R.K. Iyer. *Networked Windows NT system field failure data analysis*. in *Proceedings. 1999 Pacific Rim International Symposium on Dependable Computing, 1999.* 1999.

174. Nurmi, D., J. Brevik, and R. Wolski, *Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments* in *Euro-Par 2005 Parallel Processing*. 2005, Springer Berlin / Heidelberg.

175. Nadeem, F., R. Prodan, and T. Fahringer. *Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-purpose Grid*. in *8th IEEE International Symposium on Cluster Computing and the Grid, CCGRID '08*. 2008. Lyon, France.

176. Iosup, A., et al. *On the dynamic resource availability in grids*. in *Grid Computing, 2007 8th IEEE/ACM International Conference on*. 2007.

177. Siewiorek, D.P. and R.S. Swarz, *Reliable Computer Systems: Design and Evaluation*. 3rd ed. 1998, Natick, Mass.: A K Peters.

178. Hacker, T.J., F. Romero, and C.D. Carothers, *An Analysis of Clustered Failures on Large Supercomputing Systems.* Journal of Parallel and Distributed Computing, 2009. **69**(7): p. 652-665.

179. Platis, A., et al., *A Two-Phase Cyclic Nonhomogeneous Markov Chain Performability Evaluation by Explicit Approximate Inverses Applied to a Replicated Database System.* Journal of Mathematical Modelling and Algorithms, 2003. **2**(3): p. 235-249.

180. Koutras, V.P., A.N. Platis, and G.A. Gravvanis, *Software Rejuvenation for Resource Optimization Based on Explicit Approximate Inverse Preconditioning.* Applied Mathematics and Computation, 2007. **189**(1): p. 163-177.

181. Koutras, V.P., A.N. Platis, and G.A. Gravvanis, *On the Optimization of Free Resources Using Non-Homogeneous Markov Chain Software Rejuvenation Model.* Reliability Engineering & System Safety, 2007. **92**(12): p. 1724-1732.

182. *Software Engineering for Service-Oriented Overlay Computers (SENSORIA).* Available from: http://www.sensoria-ist.eu.

183. Young, J.W., *A First Order Approximation to the Optimum Checkpoint Interval.* Communications of the ACM, 1974. **17**(9): p. 530-531.

184. Zhang, Y., et al., *Performance Implications of Failures in Large-Scale Cluster Scheduling*, in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Editors. 2005, Springer Berlin / Heidelberg. p. 233-252.

185. Grinstead, C.M. and J.L. Snell, *Introduction to Probability*. Second Revised ed. 1997, Providence, Rhode Island: American Mathematical Society.

186. Resnick, P., et al., *Reputation Systems.* Communications of the ACM 2000. **43**(12): p. 45-48.

187. *Grid Operations Centre DataBase*. [cited 2008 01/09]; Available from: https://goc.gridops.org/.

188. *Worldwide LHC Computing Grid* Available from: http://lcg.web.cern.ch/LCG/.

189. *GOCDB/Input System User Documentation*. Available from: https://wiki.egi.eu/wiki/GOCDB/Input_System_User_Documentation.

190. *European Organization for Nuclear Research (CERN)*. Available from: http://public.web.cern.ch/public/.

191. *Grille de Recherche d'Ile de France (GRIF)*. Available from: http://www.grif.fr/.

192. *GOCDB User Documentation*. Available from: http://goc.grid.sinica.edu.tw/gocwiki/GOCDB_User_Documentation.

193. *Operations:NewNGIs creation*. 26/10/2010 Available from: https://wiki.egi.eu/wiki/Operations:NewNGIs_creation.

194. *EGEE production infrastructure: intervention procedures*. 15/05/2007 Available from: https://edms.cern.ch/document/829986.

195. *EGEE Intervention Procedures*. 08/10/2009 Available from: https://edms.cern.ch/document/1032984.

196. Rausand, M. and A. Høyland, *System Reliability Theory: Models, Statistical Methods, and Applications*. 2nd ed. 2004, Hoboken, NJ ; [Chichester]: Wiley-Interscience.

197. Bedford, T. and R. Cooke, *Probabilistic risk analysis*. 2001, Cambridge: Cambridge University Press,

198. Abernethy, R.B., *The new Weibull handbook: reliability & statistical analysis for predicting life, safety, risk, support costs, failures, and forecasting warranty claims, substantiation and accelerated testing, using Weibull, Log normal, Crow-AMSAA, Probit, and Kaplan-Meier models*. 5th ed. 2006, North Palm Beach, Fla.: R.B. Abernethy.

199. Murthy, D.N.P., M. Xie, and R. Jiang, *Weibull models*. Wiley series in probability and statistics. 2004, Hoboken, N.J. ; [Great Britain]: Wiley Interscience.

200. Lawless, J.F., *Statistical models and methods for lifetime data*. 2nd ed. ed. Wiley series in probability and statistics. 2003, Hoboken, N.J. ; [Great Britain]: Wiley-Interscience.

201. Rigdon, S.E. and A.P. Basu, *Statistical Methods for the Reliability of Repairable Systems*. 2000, New York ; Chichester: Wiley.

202. Duane, J.T., *Learning Curve Approach to Reliability Monitoring.* IEEE Transactions on Aerospace, 1964. **2**(2): p. 563-566.

203. Pukite, J. and P. Pukite, *Modeling for Reliability Analysis: Markov Modeling for Reliability, Maintainability, Safety, and Supportability Analyses of Complex Systems*. 1998: Wiley-IEEE Press

204. Howard, R.A., *Dynamic probabilistic systems*. Vol. 1. 1971, New York ; London: Wiley.

205. Siewiorek, D.P. and R.S. Swarz, *Reliable computer systems : design and evaluation*. 3rd ed. 1998, Natick, Mass.: A K Peters.

206. Jackson, S.L., *Research Methods And Statistics: A Critical Thinking Approach*. 3rd ed. 2008, Australia: Heinle Cengage Learning.

207. Silva, F.A.B.d. and H. Senger, *Improving Scalability of Bag-of-Tasks Applications Running on Master-Slave Platforms.* Parallel Comput., 2009. **35**(2): p. 57-71.

208. Neogy, S.K., et al., *Mathematical Programming and Game Theory for Decision Making*. Statistical Science and Interdisciplinary Research, ed. S.K. Pal. 2008, Singapore; Hackensack, NJ: World Scientific.

209. Onwubolu, G.C. and B.V. Babu, *New Optimization Techniques in Engineering*. Studies in fuzziness and soft computing. 2004, Berlin; New York: Springer.

210. Chen, D.-S., R.G. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. 2010, Hoboken, N. J.: John Wiley & Sons.

211. Buyya, R., M. Murshed, and D. Abramson. *A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids*. in *Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*. 2002. Las Vegas, USA.

212. Allenotor, D. and R.K. Thulasiram. *Evaluation of a Financial Option Based Pricing Model for Grid Resources Management: Simulation vs. Real Data*. in *12th IEEE International Conference on High Performance Computing and Communications (HPCC)*. 2010. Melbourne, VIC.

213. Stuer, G., K. Vanmechelen, and J. Broeckhove, *A Commodity Market Algorithm for Pricing Substitutable Grid Resources.* Future Generation Computer Systems, 2007. **23**(5): p. 688-701.

214. Cheliotis, G., C. Kenyon, and R. Buyya, *10 Lessons from Finance for Commercial Sharing of IT Resources*, in *Peer-to-peer computing : the evolution of a disruptive technology*, R. Subramanian and B.D. Goodman, Editors. 2005, Idea Group Publishiing: Hershey, PA.

215. Beasley, J.E., *Advances in Linear and Integer Programming*. Oxford lecture series in mathematics and its applications. 1996, New York; Oxford: Clarendon Press; Oxford University Press.

216. Schrijver, A., *Theory of Linear and Integer Programming*. 1998, Chichester: John Wiley.

217. *IBM ILOG CPLEX Optimizer*. Available from: http://www.ibm.com/software/integration/optimization/cplex-optimizer/.

218. *Gurobi Optimization*. Available from: http://www.gurobi.com/.

219. *GNU Linear Programming Kit*. Available from: http://www.gnu.org/software/glpk/.

220. Fourer, R., D.M. Gay, and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. 2nd ed. 2003, Pacific Grove, CA: Thomson/Brooks/Cole.

221. Banks, J., et al., *Discrete-Event System Simulation*. 5th ed, Upper Saddle River: Prentice Hall.

222. Casanova, H. *Simgrid: a Toolkit for the Simulation of Application Scheduling*. in *The First IEEE/ACM International Symposium on Cluster Computing and the Grid*. 2001. Brisbane, Australia.

223. Bell, W.H., et al., *Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies.* International Journal of High Performance Computing Applications, 2003. **17**(4): p. 403-416.

224. Iosup, A., et al., *The Performance of Bags-of-Tasks in Large-Scale Distributed Systems*, in *Proceedings of the 17th international symposium on High performance distributed computing*. 2008, ACM: Boston, MA, USA.

225. Moore, G.E., *Cramming More Components onto Integrated Circuits.* Electronics, 1965. **38**(8): p. 114-117.

226. Lublin, U. and D.G. Feitelson, *The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs.* Journal of Parallel and Distributed Computing, 2003. **63**(11): p. 1105-1122.

227. *The Grid Workloads Archive*. Available from: http://gwa.ewi.tudelft.nl.

228. *MathWorks*. Available from: http://www.mathworks.co.uk/.