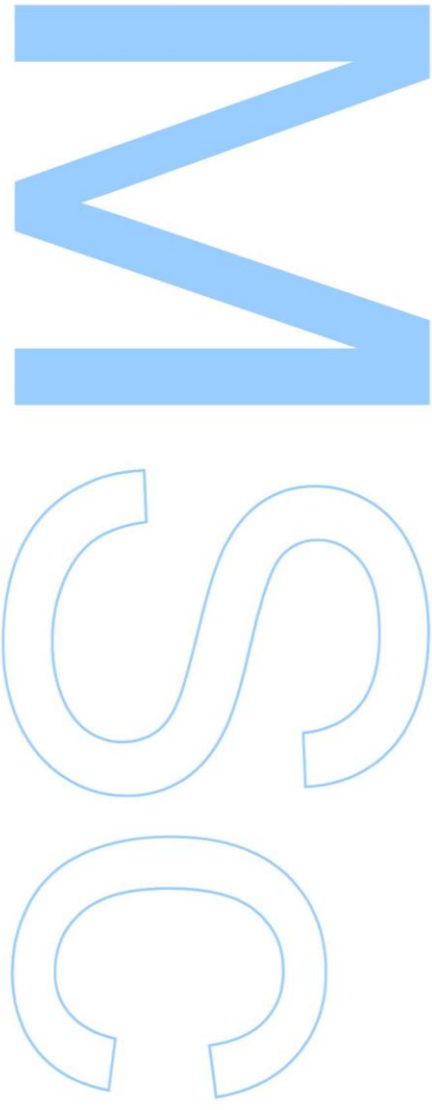
Grid Scheduling Strategies for Applications that require Data Transfer

Kiran Ali
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2014

Orientador
Inês Dutra, Prof. Auxiliar, Faculdade de Ciências



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Acknowledgments

First of all I would like to thank all mighty ALLAH for each and everything. I would like to express my special appreciation and sincere gratitude to my advisor Professor Dr. Ines Dutra, she has been a tremendous mentor. I would like to thank for her encouragement, motivation and excellent guidance. I could not have imagined a better advisor and mentor for my Master's thesis.

Besides my advisor, I would also like to thank Alpio M. Jorge for his great help in initial phases of my Masters program. I would like to thank my whole family. Words cannot express how grateful I am to my mother, father, my mother-in-law, and father-in-law for their support and prayers. At the end very special thanks to my dear husband Muhammad Ali Awan who spent sleepless nights with me and was always my support in the moments when there was no one to answer my queries. I would like to thank him for making my mind relax during the hard times of my thesis.

"Education is not the learning of facts, but the training of the mind to think."

Albert Einstein

Resumo

Computação grid é uma tecnologia que permite o compartilhamento de recursos que estão geograficamente dispersos e utilizam redes heterogêneas em ambientes de larga escala. Uma infraestrutura de grid provê serviços e funcionalidades tais como controle de instrumentos científicos, recursos, virtualização, auto-adaptação e gerenciamento uniforme. O compartilhamento de recursos em grid requer algum componente de software capaz de distribuir tarefas para as filas dos gerenciadores remotos de recursos. Apesar de existirem vários trabalhos na literatura que propõem diferentes estratégias de escalonamento e distribuição de tarefas no grid, ainda há muito trabalho a ser feito, especialmente quando se lida com aplicações que precisam analisar e processar dados que estão localizados em máquinas remotas. Normalmente, nestas aplicações, o sistema traz dados de diferentes fontes, processa-os e exibe os resultados. Em ambientes de grid, esta sequência de operações, com estes dados remotos, pode trazer vários problemas: (1) os dados solicitados pela aplicação podem não caber no espaço de armazenamento local; (2) o tempo de processamento local pode ser muito menor do que o tempo de transferência dos dados; (3) os resultados de processamento podem não caber no espaço de armazenamento local; (4) a máquina selecionada para executar a tarefa pode não estar a responder; e (5) o tempo de transferência pode ser proibitivo devido ao tamanho dos dados. Além destes problemas, que são intrínsecos a este tipo de aplicação, em ambientes de grid, a taxa de falhas de tarefas pode ser muito alta, tornando o desempenho muito ineficiente. Adicionado a este problema está o overhead imposto pelo próprio software de controle do grid, o grid middleware.

Neste trabalho estudamos o desempenho de transferências de ficheiros em ambientes de grid e propomos uma metodologia para seleção dos melhores recursos de acordo com o volume de dados da aplicação, utilizando os resultados do estudo de desempenho. Classificamos os recursos de acordo com seu comportamento dependendo dos tamanhos de dados a serem transferidos. Resultados mostram que a metodologia adotada pode reduzir a taxa de falhas deste tipo de aplicação em pelo menos 30%.

Abstract

Grid computing is a technology that enables geographically distributed resource sharing, wide-area communication and collaboration in large scale environments to provide powerful services/functionalities such as online control of scientific instruments, resource pooling, virtualization, self-adaptive systems, and unified management. Sharing resources in grid requires some software component capable of distributing jobs to the many local resource manager queues available. While there are several works in the literature that propose and experiment with different grid scheduling strategies, there is still much work to be done, specially when dealing with applications that need to analyze and process remote data. Usually, in these applications, the system brings data from different data sources, process them and display results. In grid environments, this sequence of operations on remote data can bring several problems: (1) the data required by the application may not fit in the local storage; (2) the time to process the data locally may be lower than the transfer time; (3) results may not fit in the local storage; (4) the selected machine where the data is located may be down; and (5) the transfer time may be prohibitive. Besides these problems that are intrinsic to these kinds of applications, in grid environments, the job failure rate is considerably high, making it difficult to run applications efficiently. Added to this is the overhead imposed by the grid grid middleware.

In this work, we study the performance of file transfers in a grid environment and propose a methodology for scheduling best machines according to the volume of data needed by the application, using results of the performance study. We classify machines according to their behavior when executing jobs that need files of different sizes. The results of the study show that if our methodology is adopted we can have at least 30% of reduction on the failure rate of the applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Grid computing is a technology that enables geographically distributed resource sharing, wide-area communication and collaboration in large scale environments to provide powerful services/functionalities such as online control of scientific instruments, resource pooling, virtualisation, self-adaptive systems, and unified management. In a grid environment, the resources may be supercomputers, software, storage systems, data sources or special services that belong to different institutions or departments of a single organization. Similarly different organizations may own the resources of the grid and they may have different administrative policies to access their resources.

The grid computing provides a common way of defining the policies of the resources that can be shared and used globally by the users across a virtual organization in the grid infrastructure. The users have direct access to computers, software, storage systems, data sources, special service and other resources transparently without having information about the location, operating system, administrative stuff and other details. This sharing is highly controlled by the resource management system, with resource providers and consumers defining clearly and carefully what is shared, who is allowed to share, and the conditions under which the sharing occurs. This concern for resource sharing makes a grid computing environment different from a traditional distributed computing environment as this does not deal with resource sharing and management across organizations.

There are many grid infrastructures available worldwide but large-scale production Grid infrastructures such as EGEE in Europe, the Open Science Grid (OSG) in the North America and NAREGI in Japan are providing their services and support

collaboration to many scientific and industrial applications from an increasing number of domains such as Biomedicine, Earth Sciences, Financial Simulations, Astrophysics, Fusion, Life Sciences, Multimedia, Material Sciences and High Energy Physics, among others. They provide the facility of sharing of computational and data resources within and between many different Virtual Organizations such as ample, eela, biomed, cms, atlas and alice, related to several different common interests.

In grid computing, a virtual organization (VO) is one of the fundamental concepts. A virtual organization refers to a particular organization or group of people with common scientific interests and requirements, who define resourse-sharing policies, work collaboratively with other members and/or share resources among themselves regardless of geographical location. These virtual organizations can share their resources collectively as a larger grid.

Sharing starts with the data in the form of files or databases but it is not limited to the files only. The other resources can also be shared such as software, licenses, services, sensors, networks and so on. In grid computing, the details are abstracted and resources are virtualized. The participants and users of the grid may have the membership of many real and virtual organizations. In any case a user has to follow the rules of the VO to gain its membership.

Sharing resources in grid requires a software component capable of distributing jobs to the many local resource manager queues available. While there are several works in the literature that propose and experiment with different grid scheduling strategies, there is still much work to be done, specially when dealing with applications that need to analyze and process remote data. Geospatial applications are an example. Usually, in these applications, the system brings data from different data sources, process them and display results. In grid environments, this sequence of operations on remote data can bring several problems:

- The data required by the application may not fit in the local storage;
- The time to process the data locally may be lower than the transfer time;
- Results may not fit in the local storage;
- The selected machine where the data is located may be down;
- The transfer time may be prohibitive.

Besides these problems that are intrinsic to these kinds of applications, particularly, in grid environments, the job failure rate is considerably high, making it difficult to run applications efficiently. Added to this is the overhead imposed by the grid middleware.

In this work, we present the performance of file transfers in a grid environment and propose a methodology for scheduling best machines according to the volume of data needed by the application, using results of the performance study. We classify machines according to their behavior when executing jobs that need files of different sizes. The results of the study show that if our methodology is adopted we can have at least 30% of reduction on the failure rate of the applications.

The remaining of this text is organized as follows. Chapter 2 gives a brief introduction to grid concepts and describes gLite, the middleware used in this work. Chapter 3 discusses about sources of overheads and the impact of failures in grid performance. Chapter 4 presents our methodology to select best machines according to data transfer sizes. Finally, we conclude this work with a discussion and final remarks.

Chapter 2

Grid Infrastructures

Grid computing is a technology that enables the sharing of computational resources (computers, software, data, sophisticated scientific instruments, etc) belonging to different institutions or departments of a single organization.

In a computing system, resources are subject to the use policies of the administrative domain to which they belong. These policies, in turn, define how each resource is to be used by the registered users. Once logged in, different users will be granted access to different files and will have different rights regarding the operations that can be performed over these files. In addition, the manager of the administrative domain may have defined which network connections are allowed in and out each of the computers in a local area network. These are just a few examples of the many use policies that need to be defined for the proper functioning of a computer system [6].

When we aggregate resources belonging to distinct administrative domains, each with its own local use policies and set of users, setting up global policies to coordinate the use of these resources by all registered users is not an easy task. This is one of the main issues that the grid computing technology solves. It provides a common ground for the specification of the global use policies for the resources that the different institutions want to share in the grid infrastructure. It provides a way to uniquely identify and authenticate users across a Virtual Organization (VO), a set of people with common interests that define policies and share resources with each other. Finally, it offers a multitude of services that facilitate the access and use of the resources that are shared.

The Open Grid Forum (OGF), an organisation that is responsible for establishing

standards for grid computing, defined the Open Grid Service Architecture (OGSA) which serves as a basis for the implementation of the grid systems, most known as grid "middleware".

A grid middleware is a software that "glues" together resources, users and use policies of different administrative domains providing the vision of a single integrated system. The middleware is constructed from a number of components which usually make up a toolkit. This toolkit provides client, server and development components for the management and use of the hardware, software, data and information that is shared, as well as for the proper execution of applications over these resources.

There are several grid infrastructures available worldwide. These can be part of national or regional grid initiatives. Currently, there are two very large infrastructures, one in the European block - the EGI (European Grid Infrastructure) and another one in North America - the OSG (Open Science Grid). They support several Virtual Organizations related to several different common interests. For example, biomed, alice, cms, atlas, eela, among many others. The EGI currently supports two middleware: gLite (http://glite.web.cern.ch/glite/) and OurGrid (http://www.ourgrid.org/). In this work, we will concentrate on the gLite middleware.

2.1 "Gridification"

Just like "webifying" makes an application ready to be deployed and execute on the internet, one may need to "gridify" their applications to run on a grid [36]. This process may comprise the creation of additional shell scripts to better exploit the distributed grid resources as well as to change the original source codes in order to include APIs to directly interact with grid services. In some cases, it is also necessary to stop using services and libraries that are not supported by the standard distribution of the adopted grid middleware. For example, distributed applications that use interprocess communication may need to be rewritten to make use of communication via grid files.

2.2 The gLite middleware

The gLite grid middleware is developed by the Enabling Grids for E-ScienceE (EGEE) collective efforts of different academic and research institutes as part of the Enabling

Grids for E-scienceE (EGEE) project. The gLite middleware combines various components developed in various related projects, particularly in Globus, Condor, LCG extended by EGEE developed services [28]. The gLite middleware facilitates the users with high level services for scheduling and running computational jobs, for accessing, moving and sharing big data with collaborators around the world and for obtaining information on the grid infrastructure and the grid applications [27]. The glite middleware runs over the Scientific Linux platform [1] and it is mainly written in $C++$.

Grid services based on the Open Grid Service Architecture (OGSA) specification, which follows the Service Oriented Architecture (SOA) involving a collection of services capable of communicating with one another. A service is known as a grid service if it is associated with a grid resource. In other words, the grid services are the web services with improved characteristics and services. The grid services control and manage the resource and its state in a grid environment and are accessible by more than one grid resource or vice versa.

In order to achieve the goals of the end user, the services provided by gLite middleware are expected to work in a concerted way. The gLite service can be grouped into five services groups: Access Services, Security Services, Information and Monitoring Services, Data Services and Job Management Services.

Security Services: The security services include Authentication, Authorization, and Auditing operations which enable the identification of the users, system and services and handle secure and confidential access to remote resources like worker or storage nodes. It also provides protection and monitoring to avoid exploitation of resources by malicious users or unauthorized third parties. In order to prevent users from making use of resources for which they do not have access a User-level authentication is required.

A user needs to join a Virtual Organization (VO) to be authenticated and authorized to using the grid resources. The Grid Security Infrastructure is based on hierarchic Public Key Infrastructure (PKI) and X.509 certificates. For the authentication, a user must have a X.509 certificate issued by universally trusted Certificate Authorities (CAs) and in order to access a specific grid resource, an authorization of a user is also required, which relies on the Virtual Organisation Membership Service (VOMS). The VOMS is the way gLite improves the management of authentication and authorization to the Grid resources. It allows to their own members to define different access rights to VOs resources.

Information and Monitoring Services: The information and monitoring services provide a method to access and publish information concerning the grid resources and their status. This information is important for the whole grid as it is used to discover the grid resources. Moreover, Job Monitoring Service and Network Performance Monitoring services can be built on top.

Job Management Services: The main job management services are Computing Element (CE), Accounting, Workload Management System (WMS), Job Provenance (JP) and Package Manager (PM). The Workload Management System (WMS) accepts user jobs, schedules them on available CEs according to the user preferences and several policies, keeps track of the jobs and retrieves their output.

The CE provides the virtualization of a computational resource (e.g., cluster, supercomputers or individual workstations) and it also provides information about the underlying resources. It is used as common interface to submit and manage jobs on resources. The Accounting service takes into account not only computing, but also storage and network resources.

The purpose of the Job Provenance (JP) service is to provide persistent information on jobs executed on the Grid infrastructure for later observation, data-mining operations, and possible reruns. The Package Manager (PM) service allows the dynamic deployment of application software.

Data Services: Storage Element, File & Replica Catalog Services and Data Management are three main services that are related to data and file access in grid environment. The Storage Element (SE) provides the virtualization of a storage resource, the catalog services keep track of the data location as well as relevant metadata (e.g. checksums and filesizes) and the data movement services allow for efficient managed data transfers between SEs.

The detail of the main components or services implemented by the gLite middleware is given in the next sections.

2.3 Job Description Language

When a user submits a job to a grid, it is described in a specific language, the gLite Job Description Language (JDL) and the job description file contains the characteristics and requirements of the job. The JDL is based on the classified advertisement

(ClassAds) language used to describe jobs and aggregates of jobs with arbitrary dependency relations [20]. It is also used to describe resources. It uses a semi-structured data model, so there is no specific schema required for the resource or job description, which allows it to work naturally in a heterogeneous environment.

Jobs descriptions in JDL format are text files with the syntax ***attributes = expression*** that include the command-line instruction to be executed on the computing resources as well as various parameters for resource selection and other variables related to the handling and execution of the job. A set of specific attributes are defined in gLite JDL to specify Simple, MPI-based, batch or interactive, partitionable jobs and Direct Acyclic Graphs (aggregates of jobs with dependencies) [20].

JDL can also be used to specify constraints that need to be satisfied for the selected computing and storage resources. The data access requirements can also be specified in the JDL file which are appropriate methods to define the constraints about the data and its physical/logical location within the grid. The attributes are also used in the JDL file to express the preferences for choosing among the suitable resources.

The JDL attributes are usually decomposed in the following groups.

- **Job attributes:** through which the job's specific information and actions are specified that have to be performed by the WMS to schedule the job.
- **Data attributes:** represent the job input data and storage element related information. They help in selecting the resources from which the job has the best access to data.
- **Requirements and Rank:** through this attribute, a user can specify the needs and requirements of the job's CE and preferences can also be specified in term of resources.

2.4 Computing Element

In a grid environment, the computing element (CE) is the component that deals Worker Nodes (WN) for the actual execution of a Grid job. It can represent a cluster or other grid component on which computations take place. The main function of a CE is job management. It provides the information about the underlying resource and acts as generic interface to submit and manage jobs on the resource. The CE contains

two logical parts, the gatekeeper(Grid Gate)/job manager, that enables access to the CE and a collection of WNs, which are the nodes that execute the jobs. Jobs assigned to the CE are distributed to the Worker Nodes via a Local Resource Management System (LRMS), a very important component of each CE that schedules the jobs like a normal batch system. OpenPB [46], PBSPRO [38], LSF [17], Condor [22], Torque [33] are the supported LRMS types in gLite. The gatekeeper service provides the job information to the CE via a HTTP request and the necessary user credentials are also transmitted while a secure connection is established via the Grid Security Infrastructure (GSI). The CE evaluates the given JDL file and matches the specified job requirements to its available computing resources (for examples, to only select the WN that has CPU characteristics described in the JDL file) and then assign the job to its job controller.

2.5 Workload Management System (WMS)

The workload management system composed of a set of gLite components is responsible for distributing and managing the jobs across available grid resources. The WMS basically receives job submission requests from the user Interface (UI) servers, finds the appropriate computing element for the job's execution according to the job's requirements and preferences expressed in the job description [20]. Figure 2.1 depicts the process that takes place when a job is submitted to the Grid. The individual components are described below in detail.

The selection of the resources for the job's execution is the outcome of a matchmaking process between the job submission requests and available resources. The matchmaking service is provided by the Matchmaker (MM) or Resource Broker (RB) components of the WMS service to select the resources that best match the job's requirements. The information about the resources is held by the Information Super Market (ISM) repository that is updated periodically.

Each CE contacts periodically the corresponding Berkeley Database Information Index (BDII), which is an information system for grid computing infrastructures and the BDII informs periodically the ISM. Another main component of the WMS is the Task Queue (TQ) that basically keeps the job if the required suitable resources are not immediately available that match the job's requirements. Moreover, the WMS uses the Logging and Bookkeeping service to track the job's status and after job completion

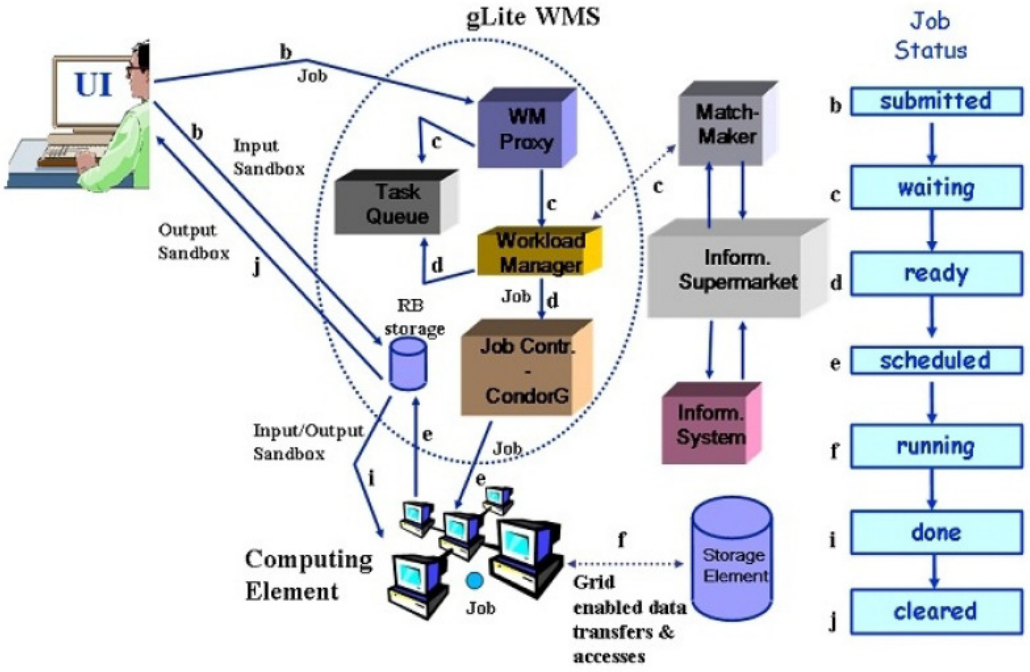


Figure 2.1: gLite Job Workflow [20]

the job it retrieves the job's results. It may require to communicate with the Storage Element (SE) if a grid file is needed by the user's job. After submitting a job, users can cancel the job, track job status or retrieve the output sandbox if the job is already successfully completed. In the WMS, the authentication and authorization occurs in the same way as with other Grid components. The proxy credentials are used for the user authentication. In addition to the initial proxy credential submission together with the job, the WMS also provides a Proxy Renewal Service that detects expiring certificate and if possible requests credential renewal from MyProxy Service.

Although the WMS performs an important role in actual execution, it does not give much support for job handling. Users have to manually keep track of jobIDs. They have to manually check the status of the job. If the job is done, they have to manually request for transferring the results back to their personal machine. This helps when the number of jobs are limited, but users may have hundreds or thousands of running jobs on the grid. In that case a more sophisticated job handling is required.

Jobs in gLite can be in one of the following states, as shown in Figure 2.2.

- **Pending:** Job is submitted from User Interface (UI) to the grid.

- **Submitted:** The one or more files specified in the job description are initially copied to the WMS when the user submits a job from the UI to the gLite WMS. The status of the job is submitted and managed by the Logging and bookkeeping ($L\&B$) service. Afterwards the specified files are copied to the Worker Node (WN) for execution.

- **Waiting:** The WMS finds the best required available CE for the job to execute. It takes information from the Information Supermarket (ISM) about the computational and storage resources and contacts with the File Catalog to access the location of any required input file. An event is logged in the ($L\&B$) and the status of the job is Waiting.

- **Ready:** The WMS assigns the job to the selected CE (but not yet transferred to CE) along with the wrapper script and other parameters. The status of the job becomes Ready in the ($L\&B$) service.

- **Scheduled:** The CE receives the job and sends the job to the Local Resource management System (LRMS). The job waits in the Computing element's queue for execution and the status of the job is Scheduled in the ($L\&B$).

- **Running:** The Input Sandbox files are copied from the gLite WMS to the available Worker Node where the job is executed. The LRMS manages the execution of the job on the Local Working Nodes. An event is logged in the ($L\&B$) and the status of the job is Running.

- **Done:** If the job is successfully completed without any error, the results are transferred back to the gLite WMS machine. The Status is updated to Done in the ($L\&B$).

- **Aborted:** If the job takes longer to finish or the proxy certificate is expired, it is aborted by the WMS. An event is logged in the ($L\&B$) and the status of the job is Aborted.

- **Canceled:** After Submitting the job the user can cancel it and the status of the job in the ($L\&B$) is Cancelled.

- **Cleared:** When the user retrieves the output files of the job to the UI, an event is logged in the ($L\&B$) and the status of the job is Cleared.

The "V" numbers in the Figure 2.2 show different events generated by the other components of gLite middleware. The ***v1=userinterface_regjob_Epoch*** is an event

that occurs when a job is submitted from the UI to the Grid. The network server of the gLite middleware accepts the job and generates an event ***v2=networkserver_accepted_Epoch***. The ***v3=workloadmanager_match_Epoch*** event occurs during the match-making process between the job and the available resource. The ***v4=jobcontroller_transfer_Epoch*** event is generated when a job is submitted to the selected CE. When the selected CE accepts the job and sends it to the local Resource Management System (LRMS) for execution an event ***v5=logmonitor_accepted_Epoch*** occurs. The events ***v6=lrms_running_Epoch*** and ***v7=logmonitor_running_Epoch*** are the events that occur when a user's files are copied from the WMS to the worker node (WN) where the job executes. When a job is completed without errors and the job's output files are transferred back to the WMS, an event ***v8=lrms_done_Epoch*** occurs. An event ***v9=logmonitor_done_Epoch*** is generated when a user retrieves his/her the output files to the UI.

2.6 Worker Node

In the gLite execution flow, Worker Nodes are the final elements, attached to a CE's batch system and receive jobs for execution. They are a set of cores managed by the CE's resource manager. A WN-local Unique account ID that is taken out of a local pool of account, is assigned (associated) to each job to create a Sandbox environment for the execution of the job. When the job is completed, the sandbox must be cleared and the job results need to be stored or transferred back to the WMS. The Worker Node(s) selected by the CE computes the job till its completion and returns the results to the CE.

2.7 Logging and Bookkeeping

The Logging and bookkeeping service ($L\&B$) is used by the WMS internally to collect/store all the information related to the job life cycle and provides an overall view of the job status to the user. This service collects events that are passed from other components such as CE and WMS, in a non-blocking asynchronous way with a robust delivery mechanism. Its operation is transparent to the user without any interaction. It implements a database schema that contains information about the job description (the JDL file), the time-stamps related to the various states of the jobs in

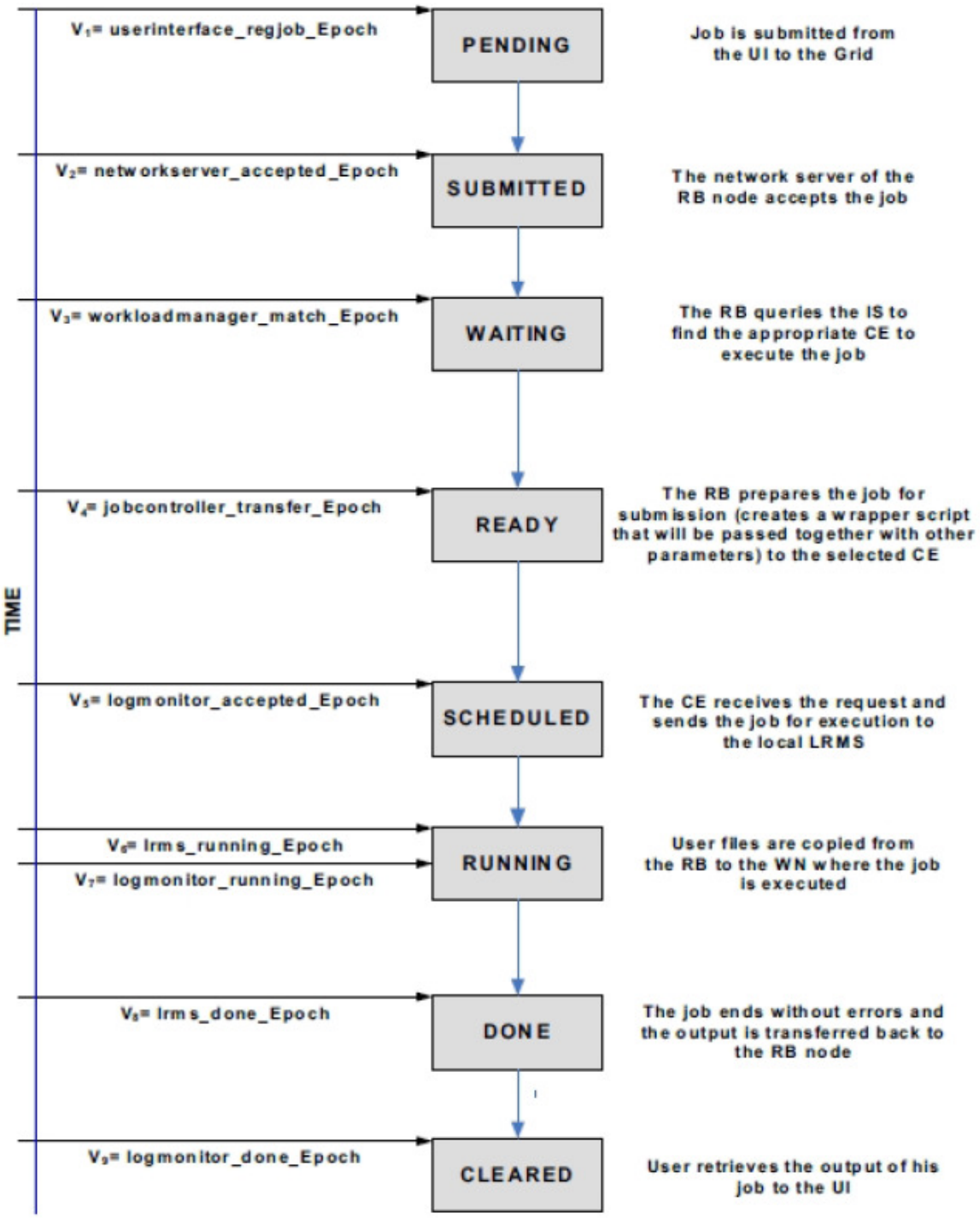


Figure 2.2: Jobs Status[14]

the system etc.

When a job is submitted, a jobID is assigned to the job and registered with *L&B*. From then on, every event related to the job is tracked by *L&B*; its architecture is job-centric. The *L&B* service can also collect the user information as well in the form of arbitrary "name = value" tags assigned to a job both from a running application or independently.

The gLite user interface commands are used to provide the job status information collected by the *L&B*. In addition to this simple querying system, a public interface (*L&B* API, available in C, C++ and java or as a web-service interface) is given to any individual Grid user who can pose simple or more complex queries for job information or register for notifications.

When the events are transformed into job states, the user can synchronously query for job information through the public interface (via HTTP or HTTPS). Moreover, every processed event is matched with a list of registrations for notification by the *L&B*. If a match is found, a notification for the corresponding status is created and sent to the registered notification listener by the *L&B*.

The security concept that the *L&B* implement is common to all gLite components. If users want to check the status information, they need to authenticate with their respective proxy credentials that allow users to share their authorization and have right to query the *L&B* for information about their jobs.

2.8 Storage Element (SE)

The Storage Element is a service that provides the virtualization of a storage resource which can vary from simple disk servers to complex hierarchical tape storage systems. This service allows a user or an application to store data for future retrieval. In SE all the data must be considered read-only (except for an application and its owner) and therefore can not be changed unless physically removed and replaced. There are different data access protocols and interfaces that Storage elements support such as: The GSIFTP (Grid Security Infrastructure File Transfer Protocol) is the protocol that provides the functionalities of FTP, but with support for GSI. It is used for secure, fast and efficient file transfers to/from storage elements.

RFIO was developed to access tape archiving systems such as CASTOR (CERN

Advanced Storage manager) and it comes in both secure and insecure version. Another protocol is gsidcap that is the GSI enabled version of the dcache native access protocol, dcap, used for local and remote file access.

The Storage Resource Manager (SRM) has been designed to be a single interface to manage the storage resources. It hides the complexity of the resources setup behind it and provides capabilities like transparent file transfer from disk to tape for specified lifetime, space reservation for new entries and so on.

Different storage elements in he Worldwide LHC Computing Grid (WLCG)/EGEE offer different version of the SRM protocol and they may also have varying capabilities. There are many types of SRM implementations in use. The Disk Pool Manager (DPM) is used for relatively small SEs with disk-based storage only. A virtual file system hides the complexity of the disk pool architecture and the secure RFIO protocol allows file access from the WAN. Like in DPM, a virtual file system (namespace) supports the transparently file transfers from disk to tape.

2.9 User Interface

In standard gLite environments, the User Interface is the access point to the gLite grid. It is a piece of software that can be installed in any machine where users have a personal account and where their user certificate is installed. From the UI, a user can be authenticated and authorized to use the EGEE resources and can access the functionalities offered by the Workload and Data management systems. It consists of a set of Command Line Interface (CLI) tools to perform some basic Grid operations and provides the information about application's characteristics, the user performance criteria and user's constraints. Following are the functionalities provided by the UI:

- Listing of all the resources suitable to run a given job according to the job requirements.
- Job submission for execution on a computing element.
- Cancel one or more submitted jobs.
- Retrieve the output files of one or more finished jobs.
- Retrieve the logging and bookkeeping information of the jobs.

Here is the short introduction of basic commands of the gLite command line interface. Before submitting the job, the user needs to have their proxy credentials. ***myproxy-init*** is used to create and upload credentials that can be used for credential renewal during the job. ***voms-proxy-init*** is executed to create the initial proxy certificate including VOMS authorization extensions which is valid for 12 hours by default and it is saved to a temporary directory with its private key to form the initial proxy credentials. A JDL file together with the proxy credential is submitted to the WMS using the command "glite-wms-job-submit" and returns a WMS job identifier (ID) for future reference, especially for status inquires. After submission, the user can check the status of a submitted job with ***glite-wms-job-status***. The user can also check the list of events that were collected over the lifetime of a given job in the Logging & Bookkeeping service by issuing the command ***glite-wms-job-logging-info***. While the job is completed, the user can retrieve its output files using the ***glite-wms-job-output***. The jobID is used as a parameter to identify one specific job. ***glite-wms-glite-cancel*** command is used to cancel one or more submitted job by the user.

2.10 Job Types

Besides just submitting a single job and waiting for it till its completion in order to get the results, we can also submit many similar jobs with different parameters or several different jobs at once which can be independent or dependent on other jobs output. The JDL allows a number of description of different job types such as Simple job, Direct Acyclic Graph, Collection, Partitionable, Parametric jobs and MPI jobs [20].

Chapter 3

Performance Evaluation of Grid Infrastructures

3.1 Sources of Overheads

Grid computing has the ability to provide coordinated resource sharing among different multi-institutional virtual organizations. The sharing is not particularly file exchange but rather direct access to computers, data, software and other resources [16]. Quality of service is a fundamental issue in the Grid. In this chapter, we present a literature review of performance of grid systems. Most work has been done to analyze the grid performance in terms of makespan (total execution time to execute all jobs belonging to an application). There are many applications that use the network to transfer data, the requirement of geospatial data processing. When a grid node accepts several requests, the overall execution performance can be significantly affected due to the overhead introduced by the Grid middleware. In fact, there are several sources of overhead associated to a grid infrastructure that can be divided into four main categories such as middleware, data transfer, loss of parallelism, and activity related overheads [39]. Figure 3.1 illustrates the hierarchy of total overheads that occur when executing scientific workflows in dynamic grid environments and could be the reason of performance losses.

- Overheads related to Middleware: The middleware overheads are introduced by the middleware services to support the proper execution and completion of

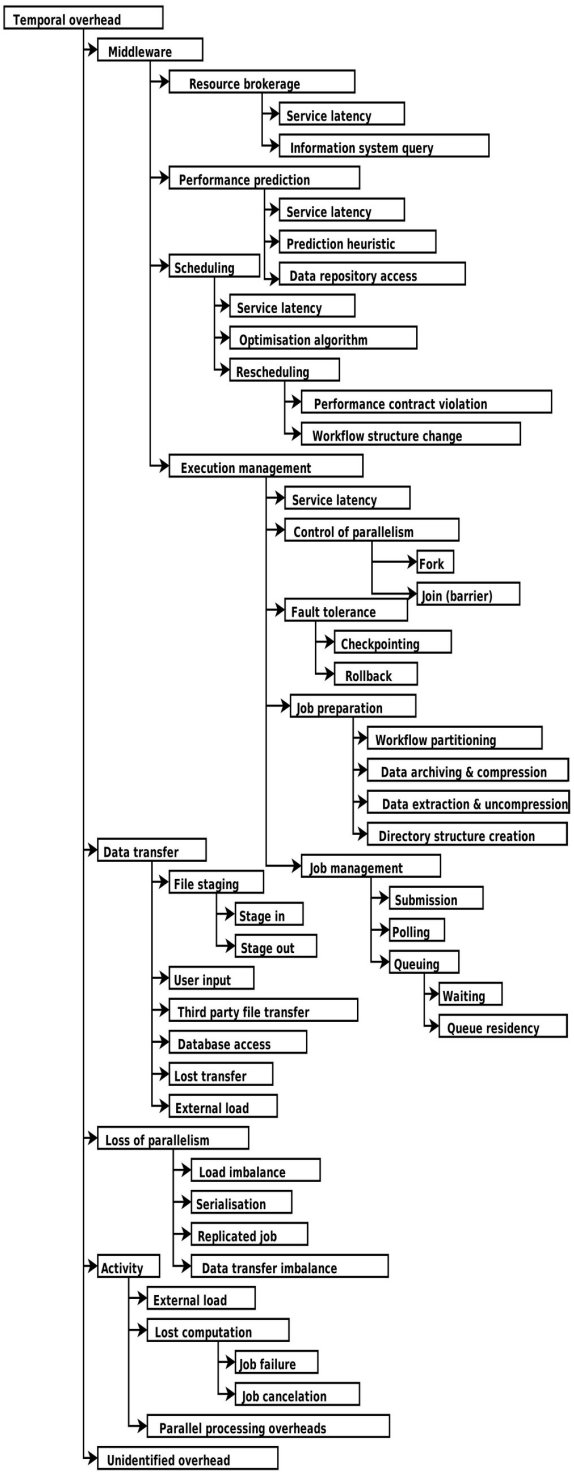


Figure 3.1: Grid workflow overhead classification[39]

the workflow. From the point of view of the middleware, the various layers of software impact performance as well as the choice of the next task or resource to execute a given application. Machine and various levels of software failure may also impact performance and communication bandwidth can become a bottleneck.

- Overheads related to Applications: From the point of view of the application, some of the factors that affect performance are related to the data representation or file representation and the application programming model.

- Overheads related to Data Transfer: The data transfer overhead occurs due to any kind of data transfer including input/output file staging between the local computer and grid site, database access, third-party file transfer upon large data dependencies, user input and unbalanced parallel data transfers (for example, due to different data sizes or network bandwidth). These overheads include any network traffic or interceptions since the wide area network is a shared resource in the grid environment.

To manage connections, dealing with the Web Services protocols and parsing the requests require a non negligible amount of computing resources [9]. Carrera et al. [9] studied the performance issues that affect the global behavior of grid services of a Grid middleware based on web services such as the Globus toolkit 3 [4]. As an example of overhead related to data representation, the distributed scientific computing applications need to exchange large arrays of floating point numbers and the representation of information in the XML language may need up to 10 times more room than the same information requires to be represented in the corresponding binary format. In order to increase the performance, the impact of representing XML data using non-text format is discussed in [42, 43, 8] as a generic technique and introduced in [21] when it is applied to the Web Services technologies.

In [13], an alternative to text compression is discussed, where the authors proposed that Simple Object Access Protocol (SOAP) must be extended to support the binary formats. In [45, 44], similar proposals are introduced. In fact, to resolve this issue, the researchers have been exploring the techniques and proposing extended versions of SOAP (the Simple Object Access Protocol used with XML files) to support the binary representation of scientific numbers. In geospatial application [37], some of the data is represented in XML files which could be a problem in future. The World Climate Global Data (`http://www.worldclim.org`) uses text files, contain binary annotation of data and the ones with current higher resolution of 30 seconds (in the

near future this will be increased to 1 second) can hold GigaBytes of space in binary representation.

Christodoulopoulos et al. [14] studied the performance of the European Grid Infrastructure (EGI) by observing the job arrival process and time duration of a job at different states. They defined four delay components of the job processing, each one corresponding to the time that a job spent at different states of its processing in the EGEE environment, from submitting a job until retrieving the corresponding output results. The time duration of the observation was one month and the total number of jobs submitted during this time period was 2,228,838. There were 343 CEs available at that time for the execution of jobs in the Enabling Grids for E-sciencE (EGEE). They evaluated the efficiency of the EGEE environment to be about 34% which means that they would have obtained the same performance if they submitted all jobs to 90 local machines. With 90 local machines all jobs would have completed within the same makespan as when executing in the grid.

Some of the overheads of a grid middleware are intrinsic to software used and the end user usually can not interfere. One alternative to reduce overheads from the user side is to implement a scheduler at application level. In the next Section, we present some of the works in the literature related to application scheduling and job scheduling.

3.2 Scheduling

More Applications are turning to Grid computing to meet their computational and data storage requirements. Single sites are simply no longer efficient for meeting the resource needs of high-end applications. A computational grid has many independent resource providers with different access policies and the diversity of those policies leads to a very complex allocation task that can not be manually handled by the users. This task does not only include searching for appropriate resources but also the coordination of the actual job execution on the selected set of resources. Therefore, an efficient and flexible Grid scheduling system is required to manage the job requests of the users and an effective Grid computing is possible, however, only if the resources are well scheduled.

Grid scheduling refers to the process of making scheduling decisions involving resources over multiple administrative domains and selecting machines appropriate for

executing a particular job. The workload of a Grid system is generated by independent users who submit their jobs over time. It is the task of the scheduling system to decide when and where a job is executed and to allocate required resources to the job. Some of the work related to different scheduling strategies in Grid environment is summarised as follows:

The work of Kaur and Singh [25] did a comparative survey analysis of scheduling algorithms for grouping the fine grained jobs in order to achieve better throughput, resource utilization and low communication time in Grid computing. They discussed various job grouping algorithms proposed in the literature for job scheduling in grid environment. They mentioned according to the dynamic job grouping-based scheduling algorithm proposed in [35], jobs are grouped together based on MIPS (Million Instructions per Second) of the available resource. The proposed job scheduling strategy takes into account: (1) the processing requirements for each job, (2) the grouping mechanism of the jobs based on processing capabilities of the available resources, (3) the dispatching of the job grouping to the suitable resource. They concluded that this model is efficient in a way that it reduces the processing time and communication time of the jobs but it does not consider the dynamic resource characteristics and also it lacks sufficiently utilization of the resource by the job grouping.

The work of [26] provides the scheduling framework for Bandwidth-Aware job Grouping based strategy, that groups the jobs according to MIPS and bandwidth-aware scheduling. This scheduling strategy takes into account the computational and communication capabilities of the resources. The priority of each resource is determined using network bottleneck bandwidth by this approach.The drawbacks of this scheduling strategy is insufficiently utilization of the resource and it is also not efficient to transfer the job.

Another approach [5] provides a similar scheduling strategy to [26]. The model sends group of jobs to the resource whose network bandwidth has highest communication or transmission rate but does not ensure that the resources with sufficient bandwidth will be able to transfer the group jobs within the required time. An Adaptive Fine-grained job algorithm proposed in [31] mainly focuses on fine-grained (lightweight) job scheduling in a grid. The problem of the algorithm is the time complexity of the algorithm that makes the job's preprocessing scheduling time high. Moreover, it does not focus on the memory requirement of file-size. The work of Soni et al. [24] performs the grouping on the basis of processing capabilities, memory size and the bandwidth of the available resources. The Heap sort tree is used to select the

highest computational power resource to balance the effective job scheduling.

Another interesting memory aware job scheduling model proposed by Mishra et al. [34] studied the computational-communication, memory size based job grouping scheduling strategy. The utilization of the grid resource is maximized while the network delay to schedule the job is minimized. Another job grouping scheduling approach proposed by Sharma et al. [40] improves the resource scheduling by maximizing the resource utilization and minimizing the processing time of job through a model composed of three levels called user level, global level and local level (cluster level). A huge collection of data generated by scientific instruments are stored or replicated on distributed resource to increase the storage capacity or efficiency of access. Therefore, the distributed data and computational resources can be accessed transparently by the scientists.

Venugopal et al. [47] presented and developed the Gridbus Broker, an extension of the Nimrod-G [3] computational Grid resource broker and provides services relevant to data-intensive computations. The Nimrod-G works on the optimization of the user-supplied parameters such as deadline and budget [7] for computational jobs only. It has no function for accessing remote data repositories and for optimizing on data transfer.

The Gridbus broker supports a declarative and dynamic parametric programming model for creating grid applications such as Belle Analysis Software Framework, a physics analysis application. The Grid broker works on the principle of discovering appropriate computational and data resources. It schedules the jobs based on optimization of data transfer on the suitable resources. It monitors the job execution on the selected resources and returns the results back to the user when they are finished. The Grid broker has the ability to locate and access the required data from best data repositories from multiple sites according to the availability of files and the quality of data transfer. The authors analysed three scheduling scenarios, (1) scheduling limited to only those resources which hosted the data files for the job, (2) scheduling without regarding the location of the data, and (3) adaptive scheduling proposed by the authors, optimises computation according to the location of data. They considered a data-set of 20 jobs. As there was no data transfer involved, according to the first strategy, all of the resources successfully executed the 20 jobs each. According to the second scheduling strategy, regardless of the location of the data, involves maximum amount of data transfer which causes a problem for the applications requiring large data transfers and utilising resources with slow network. In the last evaluation, the authors chose best available computing resource that had best available bandwidth

to the data host for the related data for the job. Their scheduling strategy is similar to our proposed scheduling strategies but our proposed scheduling strategies take size of the data files into account and selects the best resource accordingly for the job of the corresponding data-set. For each data-set, we classify the resources into different categories according to their performance for different particular data-set.

The work on AppLes project [11, 12, 10] investigates the adaptive scheduling for grid computing and demonstrates their effectiveness on real applications in production environments to improve the performance experienced by end-users. They considered the static and dynamic resource information, performance predictions, application and user-specific information while developing scheduling techniques to enhance the application performance. It provides logistics of the deployment that involves "*discovering resources, performing application data transfers, keeping track of the application data, launching and monitoring computations on Grid resources, and detecting and recovering from failures*"[10]. Their scheduling algorithms increase the performance by making decisions related to application data transfer/download and selection of computing element to start application tasks.

To evaluate and select the QoS guaranteed resources from a potential Grid resources for the users, Wang and Cao [48] proposed the committee-based resource evaluation and selection method denoted as CRESM. It is composed of two layers called a representative layer and a committee layer. The representative layer collects the information about the user experience about any particular grid resource. The committee layer exploits this information based on individual judgments and decides on grid selection. The fuzzy k-nearest classifier, a recognition method to classify unknown entities including neural networks is chosen to classify the information due to its reliability, stability, extensible nature and low overhead feature. The resources are divided into two classes, reliable ones which provide high QoS and unreliable ones that gives low QoS. Their results show that CRESM is stable and can accurately evaluate the reliability of the resources. CRESM not only brings QoS improvement but also increases the resource utilization ratio in the Grid. Moreover, it helps the users avoid to reserve an over estimated time on the resources.

It has been observed that the agreement based resource management can resolve the issues of dealing with policies and objectives of the different resource providers and the resource users as it provides reliable interaction between them (resource providers and users). This model needs negotiations to create bi-lateral agreements between grid parties. Li and Yahyapour [30] presented and evaluated a strategic negotiation model for grid computing which is based on utilities functions or preference relationship for

the negotiation parties and learning-based negotiation strategies. According to this model once an agreement is established, it will be committed by both parties, User and Resource provider and will not be violated.

Their negotiation model is based on bilateral negotiation model, while a Q-learning [49] algorithm was chosen as a learning based negotiation strategy. Q-learning based strategy is an online algorithm and hence well suited for dynamic and unpredictable grid environments. They used 5000 jobs from the Cornell Theory Center (CTC) workload traces [2] to do the experiments and five different simulation cases are considered. From the results, they show that learning-based negotiation model is flexible and can be applied in the practical use in automatic job scheduling and it also indicates that negotiation overhead due to exchanged messages is manageable for practical applications.

Another work related to grid scheduling is a scheduling algorithm called Multiple Queues with Duplication (MQD) for bag-of-tasks applications in grid environments, presented by Lee and Zomaya [29]. This algorithm makes scheduling decisions based on the recent workload pattern of the resources and using multiple queues and task duplication in order to gain better resource utilization. The performance of MQD algorithm was compared between previously proposed performance information dependent algorithms (PIDA), Min-Min, Max-Min and Sufferage [23, 32] and a performance information independent algorithm RR (round-robin) [18]. From the simulation results, it is noticed that when the error in performance prediction increases, the performance of the PIDAs significantly decreases.

Another scheduling strategy for grid resource allocation is Reinforcement learning (RL) proposed by Galstyan et al. [19] and implemented by Costa et al. [15] to actual grid environment. Reinforcement learning is an interesting and simple adaptive technique that may work well in actual grid environments. In RL an agent, for example a grid user, learns optimal actions through a trial and error exploration of the environment and by receiving rewards for its actions. The reward (utility) function defines what the good and bad actions are in different situations. The agent's goal is to maximize the total reward it receives. The grid users are modeled as agents and they have no prior knowledge about the capabilities of the resources. Instead, they utilize a simple reinforcement learning scheme to estimate the efficiency of different resources based on their past experience. An agent assigns a "score" that indicates how well that resource has performed in the past. After each submitted job, the agent updates the score of the corresponding resource.

Silva et al. [41] proposed a Workqueue with Replication(WQR) scheduling algorithm which delivers good performance without using any kind of information about the grid resources or tasks. The WQR is similar to the classic Workqueue but it uses task replication. In the proposed approach when a task is replicated, the completion of first replica is considered as the valid execution of the task and the other replicas are cancelled to make the resources available. However, this approach requires an additional increase in resource consumption which can be controlled by limiting replication.

Most scheduling solutions for grid environments mentioned so far focus on executing jobs without taking into consideration data transfers with the exception of Venugopal et al.'s Gridbus broker, and Casanova and Berman's APPleS work.

Another issue that is usually neglected is how to avoid selecting machines that are prone to failure. Zeinalipour-Yazti et al. [50] worked on a framework to characterize failures in a grid system. Besides characterizing the failures, it would be nice, if we are able to avoid selecting these machines to execute jobs. Ideally, it would be interesting to predict which machines have higher chance of executing jobs according to the amount of data they need to transfer from or to a data server. Our proposed strategies work on the principle of avoiding those machines that have high failure chances to execute jobs and selecting good machines for the applications that require data transfers according to the size of data.

Chapter 4

Scheduling Strategies for Applications that use File Transfers

This chapter describes a methodology and strategies to allow better performance of applications that make use of file transfers. We used a grid infrastructure, the EGI to apply our methodology and create our scheduling strategies. We used the Biomed Virtual Organization to submit all jobs from the GridUP user interface, `ui01.up.pt`. The Biomed VO is supported by many different sites, which provides access to tens of thousands of CPU cores to its users. The list of the national Grid initiatives that support Biomed by providing computing and storage facilities and technical support, is Bulgaria, Croatia, Cyprus, France, Germany, Greece, Italy, Macedonia, Moldavia, Netherlands, Portugal, Poland, Russia, Slovakia, Spain, Turkey, Ukrain, United Kingdom, Canada, Asia Pacific.

Totally depending on a grid middleware, like gLite, to execute the jobs that require data transfers can be very inefficient. The successful execution of these jobs depends on a good choice of machines which can reduce the jobs failure rate. To achieve our goal, the profiling technique is used to classify the grid machines into two categories, fully responsive (Good machines) and limited-responsive machines (Bad machines). The profiling method runs different jobs on these machines to get the aforementioned classification. Based on the experimental results and this classification, we proposed a scheduler that selects resources according to the size of the data transfers. This process can be automated and easily performed with low complexity.

4.1 Methodology

The application used in our methodology fetches text files that contain binary annotations of temperature around the world from the Wold Climate Global Data website (http://www.worldclim.org). The reason for choosing this data is because it is available in different resolution sizes and is used by geospatial applications that fetch this data for processing and analysis. The data is stored on a website in four different resolutions: data collected in an interval of 10 minutes, 5 minutes, 2.5 minutes and 30 seconds. The size of the files corresponding to each of these resolutions is given in Table 4.1. Each instance of our application is called a job. Each job has the basic task of an open geospatial system, i.e., to fetch the files from the website http://www.worldclim.org using WGET, unzip it and process it. It basically consists of a python program that fetches binary files, decodes the binary format and generates 12 comma-separated-value (CSV) files that contain monthly mean temperatures in the world. The zipped file contains 12 * 2 files, one per each month. One of the two files contains the data in binary format. The second one contains the data header. Each generated CSV contains temperatures related to a given month. Size of the binary and of the CSV files vary according to the data resolution (30 seconds, 2.5, 5 and 10 minutes). The 10 Min resolution generates CSV files of 10 MBytes, the 5 Min data-set generates CSV files of 45 MBytes and the 2.5 Min data-set has CSV files of 180 MBytes. The average runtime for this processing, excluding file transfer, on a machine, for example, "svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d", for the 10 Min resolution data-set, is 812 seconds. Similarly, for data-set 5 Min and 2.5 Min the processing took 3058 seconds and 12054 seconds, respectively, on machine "svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d".

We determined all the machines/resources/computing-elements available in Biomed VO. The words machines, resources and computing elements are used interchangeably throughout this chapter.

Resolution	Size (MBytes)
10 Minutes	6.6
5 Minutes	22.5
2.5 Minutes	79
30 Seconds	1435

Table 4.1: File sizes of different data-sets

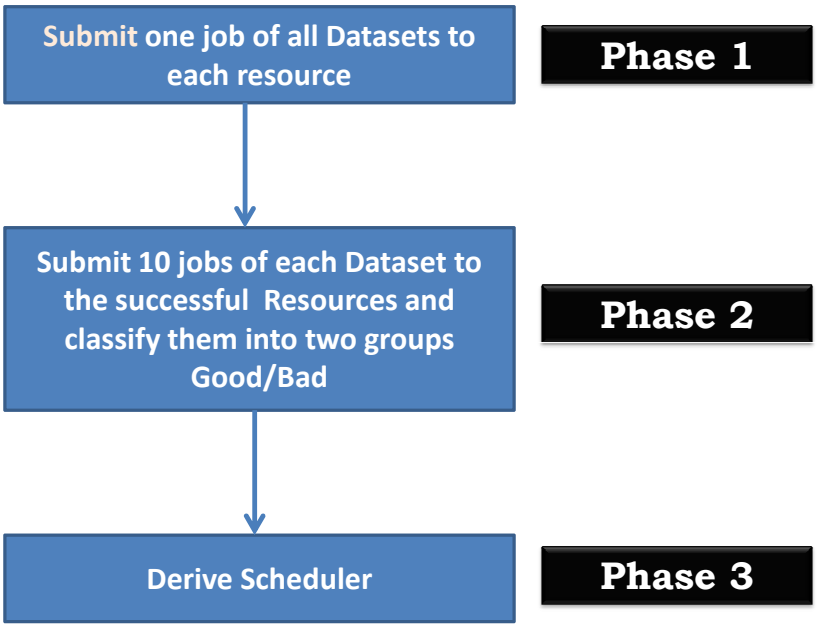


Figure 4.1: Methodology of our Approach

The total number of the CE available in Biomed VO is equal to 188. Each one of them has many processors/cores.

The methodology used in this chapter can be divided into three phases. These phases are illustrated in Figure 4.1. All of these phases are described as follows.

4.1.1 Phase 1

In the first phase, we submit four jobs corresponding to each data-set (10 minutes, 5 minutes, 2.5 minutes and 30 seconds) to each machine available in the Biomed VO. This gives us a total of $188 * 4 = 752$ jobs submitted. The JDL script used to define the job in our system is given in Figure 4.2 for reference. The jobs are submitted with the GridUP user interface `ui01.up.pt`. The results are collected from the WMS. A job is declared successful if it executes without any error. The errors are categorized into two classes:

1. the job finishes with a status Done and an exit code different from zero. This means that the job terminated, but some failure occurred in the middleware (such as a site misconfiguration problem, or a version of python not compatible)

2. the job finishes with a status Done and an exit code zero, but did not terminate

```
Type = "Job";
JobType = "Normal";
Executable = "/usr/bin/python";
StdOutput = "DownloadWorldClim.out";
StdError = "DownloadWorldClim.err";
InputSandbox = {"/home/kiran/GIS/IO.py","/home/kiran/GIS/ntplib.py"};
OutputSandbox = {"DownloadWorldClim.err","DownloadWorldClim.out","*.csv"};
Arguments = "IO.py http://biogeo.ucdavis.edu/data/climate/worldclim/1_4
            /grid/cur/tmin_10m_bil";
MyProxyServer = "px01.ncg.ingrid.pt";
ShallowRetryCount = 3;
```

Figure 4.2: JDL script used for 10 minutes samples

properly (for example, it is aborted because the proxy expired).

In this first phase, we want to rule out machines that are not responding or can not run the jobs for some reason. For example, a machine may fail execution of a job because: (1) it may not have the python version or libraries we need, (2) may not be responding or (3) may not be properly configured.

4.1.2 Phase 2

In the second phase, having ruled out all machines that failed, we will check the robustness of the successful machines by submitting various jobs of the same data-set size to these machines. We then select all machines that succeeded executing the jobs in the first phase and ignore all those machines which failed to run any of the four data-sets. For each data-set, we select all the machines that successfully executed that particular data-set in the first phase. For example, if machines M1 and M2 successfully executed the 5 minute data-set then we submit a batch of 10 jobs of 5 minute resolution to both M1 and M2.

Similarly, the other successful machines for the other data-sets are selected and they are tested with a batch of 10 jobs of the same data-set. When the jobs are

completed, we retrieve the output files from the WMS. The machines are collected per each data-set size and further divided into two groups: 1) Good machines and 2) Bad machines. The purpose of the further evaluation of machines for each data-set size is to perform the classification of machines according to their performance while executing each data-set. The machines that managed to successfully execute at least a single job are declared as Good machines for that particular data-set.

4.1.3 Phase 3

Based on the results obtained with the first two phases, we are ready to create strategies that can avoid selecting Bad machines to execute our jobs. We have proposed two different scheduling algorithms based on the results of the previous phases to rank the available machines in the grid to increase the efficiency. After ranking, we choose the best ranked machine according to the data size. The names of these scheduling algorithms are presented as follows.

- Class based Scheduling Algorithm (CBSA)
- Global Scheduling Algorithm (GSA)

Each algorithm has two versions: offline and online. The offline version is used to build the initial machine ranking. The online version is used during execution and can modify the original rank according to the machines' dynamic behaviour.

4.1.3.1 Class Based Scheduling Algorithm (CBSA)

In the class based scheduling algorithm (CBSA), we initially create clusters based on the results of the first two phases. Assume there are n data-sets used to determine the good or bad machines. We create $n+1$ clusters numbered from 1 to $n+1$. In our case there are 4 data-sets and hence the number of clusters will be 5. A machine in the grid that is good for p data-sets is added into a cluster $n-p+1$. For example, a machine which is good for all the data-sets will be added into cluster number 1. All the machines in the grid are added to their corresponding clusters. Afterwards, machines inside the cluster are ranked. The ranking inside cluster is performed based

Algorithm 1 Class based Scheduling Algorithm

1: **Offline Phase: Rank Reliable Machines**

2: Assume there are n different data-sets

3: Create $n+1$ clusters, numbered from 1 to $n+1$

4: A machine that successfully executes p data-sets will be allocated to $n-p+1$

5: i) Rank the machines within each cluster by an descending order of κ,
where $\kappa = \dfrac{\text{Total Execution Time of all Successful Jobs}}{\text{Total Number of Jobs}}$
OR
ii) Create table for each data-set inside each cluster and rank the machines in each table with respect to the number of jobs executed in phase 2 of that specific data-set

6: The ranking of the machines in this phase starts from the first machine in cluster 1 and ends with the last machine in group $n+1$

7: **Online Phase: Dynamically Updating the Ranking of the Machines**

8: Assume a machine M_i under consideration and the current cluster number of M_i is h

9: **if** (A machine M_i is unsuccessful for x times (where $x \geq 1$) && its current cluster number $h \neq n+1$) **then**

10: Demote a machine M_i from its current cluster number h to next cluster number $h+1$,

11: **else if** (A machine M_i successful for y times (where $y \geq 1$) && its current cluster number $h \neq 1$) **then**

12: Promote a machine M_i from its current cluster number h to $h-1$

13: **end if**

on a metric called κ. It is defined as follows.

$$\kappa = \frac{\text{Total Execution Time of all Successful Jobs}}{\text{Total Number of Jobs}} \tag{4.1}$$

The total execution time of successful jobs is divided by the total number of jobs.

All the machines inside each cluster are sorted in ascending order with respect to their corresponding values of κ.

Similarly, we also propose another method to rank the machines inside the clusters as well. The machines are ranked differently for each data-set in each cluster. The specific data-set will consult the corresponding ranking of its data-set inside the cluster. Assume, there are 4 data-sets and we are interested in cluster number 1. The proposed algorithm creates four tables inside cluster number 1 for each data-set. The ranking of the machines in each table is performed based on the number of successful jobs executed in the previous phase for that specific data-set. Hence, if the machine needs to be selected for a data-set b, all the machines inside the clusters corresponding to data-set b are selected to give the overall ranking.

Irrespective to the method used to sort the machines inside a cluster, the overall ranking of the machines starts from the first machine in the first cluster to the last machine in the $n+1$ cluster. The machines in the last cluster $n+1$ can be ordered with respect to their indexes or alphabetical orders of their names.

In the online phase, if a machine M_i is successful for x times then it is promoted to the next best cluster, i.e., its current cluster number minus 1. The value of x can be set to any integer greater or equal to 1. Similarly, if a machine M_i has unsuccessfully executed the y jobs then it is demoted by one cluster, i.e., its current cluster number plus 1. However, it is obvious, if the machine is in cluster number 1 then it cannot be promoted anymore and similarly, a machine in a last cluster $n+1$ cannot be further demoted. We do not remove the machines from the cluster based on the bad performance but we keep them in $n+1$ cluster. This decision is made on the assumption that some of the machines can perform well during in future. The pseudo-code presented in Algorithm 1 shows these steps.

4.1.3.2 Global Scheduling Algorithm(GSA)

The global scheduling algorithm (GSA) globally ranks the machines based on the results given in phase 2 of our methodology. We run total number of α_i jobs on each machine M_i for all data-sets. Assume α_i^j is the total number of jobs of j^{th} data-set executed on M_i. Hence, $\alpha_i = \sum_{j=1}^{n} \alpha_i^j$, where n is the total number of data-sets. Assume β_i^j and γ_i^j are the successful and unsuccessful jobs of j^{th} data-set on machine M_i respectively. The execution time of the successful jobs can be easily computed from

Algorithm 2 Global Scheduling Algorithm

1: **Offline Phase: Global Ranking of Machines**

2: Assume there are n different data-sets. Now consider a machine M_i. Let β_i^j and γ_i^j be the number of successful and unsuccessful jobs on machine M_i respectively for data-set j. Total number of jobs submitted on M_i is equal to $\alpha_i = \sum_{j=1}^{n} \alpha_i^j$. Similarly, the total number of jobs of data-set j is equal to $\alpha_i^j = \beta_i^j + \gamma_i^j$

3: A penalty of unsuccessful job execution is U time units, where U is a very big number.

4: For each machine M_i, and each data-set j
compute $\lambda_i^j = \dfrac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j}$

5: Rank all the machines in ascending order with respect to their corresponding value of λ_i^j for each data-set j

6: Ties are broken with respect to their indexes

7: **Online Phase: Dynamically Updating the Ranking of the Machines**

8: Assume a machine M_i has completed a job k of data-set j

9: **if** (A machine M_i has unsuccessfully executed a job k of data-set j) **then**

10: $\quad \gamma_i^j = \gamma_i^j + 1$

11: **else if** (A machine M_i has successfully executed a job k of data-set j) **then**

12: $\quad \beta_i^j = \beta_i^j + 1$

13: **end if**

14: $\alpha_i^j = \alpha_i^j + 1$

15: Update $\lambda_i^j = \dfrac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j}$ of M_i

16: Re-rank this machine for data-set j

their results. However, the unsuccessful jobs should be penalized with extra time. In order to do so, we assume a job that is unsuccessful on a machine has taken a time of U time units. The value of U can be set to any arbitrary big number greater than the successful job execution time. We compute a metric λ_i^j given in Equation 4.2 for each data-set j on each machine. For each data-set j, all the machines in the grid are ranked in ascending order with respect to their λ_i^j values. If two machines have the

same λ_i^j, their ties are broken based on their indexes. This procedure is repeated for each data-set and finally, we have a global list of machines ranked with respect to λ_i^j for each data-set.

$$\lambda_i^j = \frac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j} \tag{4.2}$$

In the online phase of this scheduling algorithm, the ranking of each machine is dynamically changed based on its performance. Assume a machine successfully executed a job of data-set j then its λ_i^j is re-evaluated by adding the execution of this new job and its ranking in the corresponding data-set list is updated accordingly. This procedure is performed to penalize a machine if it has unsuccessfully executed a job. The reevaluation of λ_i^j may be expensive to perform on completion of each job. This problem can be solved by storing the results of x jobs of data-set j and then the value of λ_i^j can be collectively updated for its x jobs. Please note that the value of $x \geq 1$. The pseudo-code of GSA is presented in Algorithm 2.

```
#!/bin/bash
lcg-infosites --vo biomed ce >> resources.out
sed '1,2d' -i resources.out
cut -f6 resources.out  >> onlyres.out

while read line
do
array+=("$line")
done < onlyres.out

for ((i=0; i < ${#array[*]}; i++))
do
glite-wms-job-submit -a  -o jobid -r ${array[i]} interval_10m.jdl
done >> jobs_submission_status.out
```

Figure 4.3: Shell script used to submit the jobs

4.2 Results obtained in Phase 1

As mentioned previously, we have used the VO Biomed and the EGEE Grid infrastructure. The jobs are submitted from the GridUP user interface, `ui01.up.pt`. The jobs were defined using a JDL script which invokes a python program, collects timings from the execution phase (start, start fetching file, start processing, end processing) presented in Figure 4.2. We used a shell script that selects the resource and submits the jobs of each one of the data-sets as given in Figure 4.3.

In the first phase, we submitted a job of each data-set resolutions on all machines available in the grid. While submitting the jobs of 10 minute and 2.5 minute resolutions, the grid infrastructure has 188 machines available. However, at the time of submission of 5 Min and 30 seconds data-set jobs, it has 185 and 184 machines, respectively. A subset of the machines that presents the results corresponding to different machines for different data-sets in the first phase are available in Table 4.2. A complete table is presented in Table A.1 for reference.

A machine that successfully executes the job in phase one is marked as *Successful*, while the the machine failed to execute the jobs is represented as *Failed*. For example, the machine at number 83 in Table 4.2 (*cream ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed*) has successfully executed the 10 and 2.5 Min data-set job, while failed to executed the 5 Min data-set in the first phase. Please note that the column corresponding to the 30 second data-set is omitted in the tables. The job corresponding to the 30 second data-set in the first phase failed on all the machines available in the grid infrastructure because the output sandbox in grid infrastructures has a limit in size, we can not fetch files whose total size exceeds the size limit imposed by the grid.

No	Machines In the Grid	10 Min	5 Min	2.5 Min
1	arc-ce01.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
2	arc-ce02.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
3	arc-ce03.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
4	cale.uniandes.edu.co:8443/cream-pbs-biomed	Successful	Successful	Successful
5	cccreamceli09.in2p3.fr:8443/cream-sge-long	Failed	Failed	Failed
⋮	⋮	⋮	⋮	⋮
79	cream-ce02.cat.cbpf.br:8443/cream-pbs-biomed	Failed	Successful	Failed
80	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Successful	Successful	Successful
81	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Successful	Successful	Successful
82	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Successful	Successful	Successful
83	cream-ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Failed	Successful
84	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Successful	Successful
85	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Successful	Successful	Successful

86	cream.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Failed	Successful	Failed
⋮	⋮	⋮	⋮	⋮
185	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
186	t2-ce-01.to.infn.it:8443/cream-pbs-biomed	Failed	Failed	Failed
187	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Successful	Successful	Successful
188	wario.univ-lille1.fr:8443/cream-pbs-biomed	Failed	Failed	Failed

Table 4.2: Successful and Failed Machines while Running One Job of each Data-set (Phase 1)

Data-set	No of Machines Used	Total Submitted Jobs	Successful	Failed	% of Successful Machines
10 Min	188	188	89	99	47%
5 Min	185	185	96	89	51.8%
2.5 Min	188	188	84	104	44.6%
30 Sec	184	184	0	0	0%

Table 4.3: Summary of First Phase Results

In the first phase, for the 10 Min resolution data-set, out of 188 jobs (submitted to 188 machines) only 89 jobs successfully executed the jobs and 99 jobs failed. It means that only 47% of the machines were successful. Similarly, for the 2.5 Min resolution data-set, 84 jobs were successful and 104 failed. The percentage of the successful machines is equal to 44.6%. As the number of machines available for the 5 minutes resolution data-set was 185, out of 185 submitted jobs, 96 jobs successfully executed and 89 failed. In this case, the percentage of successful machines is 51.89%.

For 30 second resolution, 184 machines were available in the grid infrastructure and one of them was successful. So the successful percentage is zero. These results are summarized in Table 4.3. As we can observe there is an oddity in the results of the first phase. The percentage of the successful machines for the 5 Min resolution data-set is better than for the 10 Min resolution data-set. This can occur in a grid infrastructure as the results depend on the workload of the machines.

4.3 Results obtained in Phase 2

We also generated tables for each data-set to categorize the Good and Bad machines determined in phase 2. A subset of those machines corresponding to the 10 Min resolution data-set is presented in Table 4.4. The complete list for each data-set

generated in phase 2 is presented in Appendix A (in Tables Table A.2,Table A.3 and Table A.4) for reference.

We analysed the percentage of the successful machines in the second phase also. The summary of the results is presented in Table 4.5. We describe the results for the 10 minutes resolution data-set here and results of the other data-sets can be easily interpreted from the given tables. For the 10 minutes resolution, 89 machines were successful in the first phase. We submitted a batch of 10 jobs of the 10 minutes resolution data-set on each of these machines. Therefore, a total of 890 jobs were submitted to these 89 selected machines. Among those 890 jobs, 661 jobs successfully completed their execution, while 229 jobs failed to complete their execution. This gives us a success rate of 74%, which is a very good improvement when compared with the experiment with the 10 minutes resolution data-set during the first phase (47%).

A machine that successfully executed all the 10 jobs submitted to it is declared as a Good Machine. We computed the percentage of the successful machines and it is 76% for the 10 minutes data-set. Similarly, the percentage of the successful jobs in this phase is equal to 74%. There are two oddities in this table. The percentage of the successful machines of 2.5 minutes resolution data-set is greater than other data-sets. Moreover, the percentage of the successful jobs of the 2.5 minutes resolution data-set is greater than the 5 minutes resolution data-set. This is perfectly acceptable in a dynamic and heterogeneous infrastructure such as a grid.

No	Machines In the Grid	10 Min
1	cale.uniandes.edu.co:8443/cream-pbs-biomed	Good
2	ce-01.roma3.infn.it:8443/cream-pbs-grid	Good
3	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
4	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Good
5	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
⋮	⋮	⋮
37	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
38	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
39	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
40	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Good
41	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
42	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
43	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
⋮	⋮	⋮
86	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
87	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
88	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
89	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Good

Table 4.4: Good and Bad Machines for the 10 Min Data-set (Phase 2)

Data-set	Successful Machines in 1^{st} Phase	Good Machines in $2^{nd} Phase$	Total Submitted Jobs	Successful Jobs	Failed Jobs	% of Good Machines	% of Successful Jobs
10 Min	89	68	890	661	229	76%	74%
5 Min	96	69	960	562	398	71.8%	58.5%
2.5 Min	84	61	840	561	279	72.6%	66.7%
30 Sec	0	0	0	0	0	0%	0%

Table 4.5: Summary of Second Phase Results

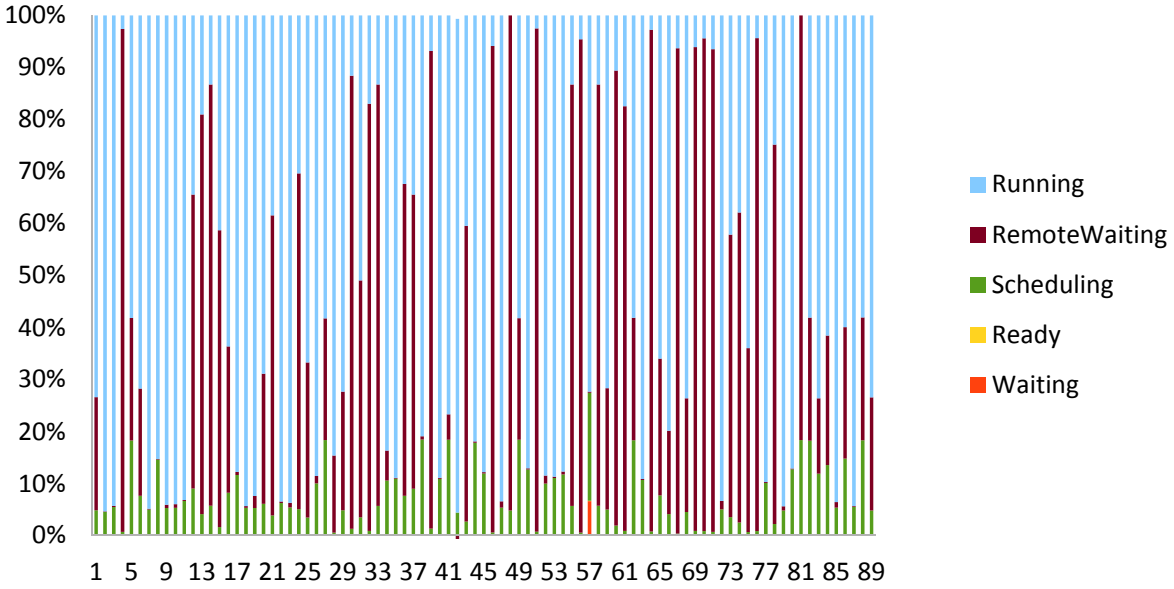


Figure 4.4: Execution Time Breakdown 10 Min (Phase 1)

4.3.1 Successful jobs

Figures 4.4, 4.5 and 4.6 present percentage of execution time breakdown of successful jobs for the 10 Min, 5 Min and 2.5 Min data-sets, respectively, in phase one. Figures 4.7, 4.8 and 4.9 show the percentage of execution time breakdown of jobs of data-sets 10 Min, 5 Min and 2.5 Min, which completed their execution in phase 2. In the Figures, it is noticeable that some of the jobs have their very high remote waiting time as compared to their execution time on the computing resource.

Figures 4.10, 4.11 and 4.12 show the total execution time per lot of 10 jobs per each data-set i.e, 10 Min, 5 Min, 2.5 Min, respectively, on each selected machine in phase 2 of the methodology. As it has mentioned previously, in the second phase we submitted 10 jobs of each particular data-set to selected machines from phase 1. In Figure 4.10, there is a total of 89 successful machines and the total execution time of 10 jobs of data-set 10 Min per machine is shown in the Figure 4.10. In these figures some of the bars show very low total execution time for some machines for all 10 jobs which corresponds to either jobs failure on those machines or that those machines are fast enough to complete all jobs in very low time. It could also be the case that some of the machines managed to execute very few jobs out of 10 jobs and have their low execution time.

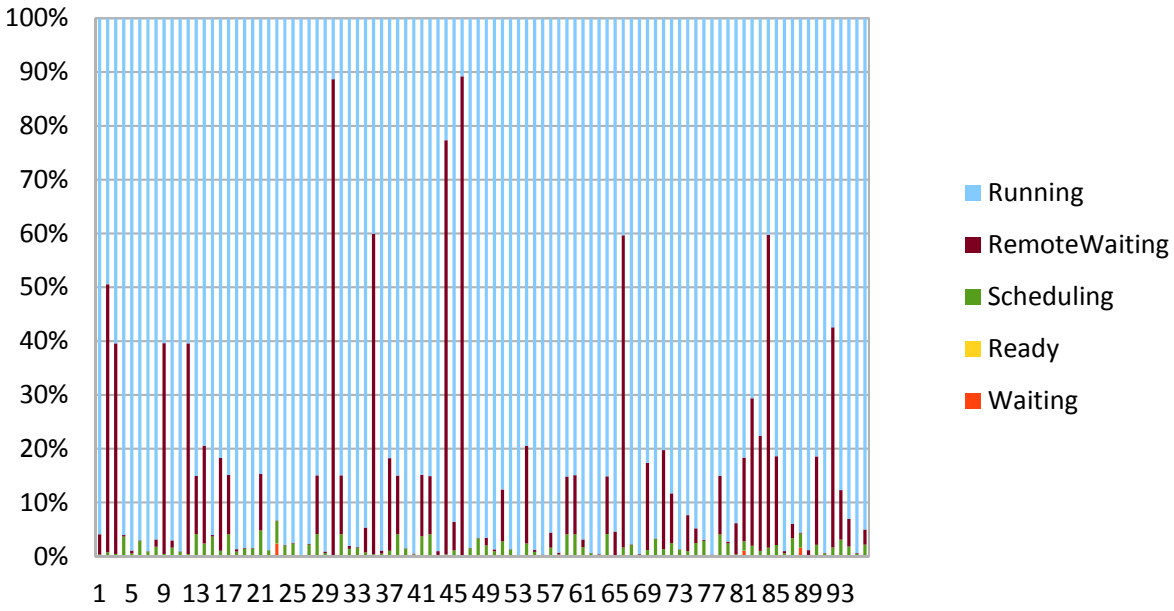


Figure 4.5: Execution Time Breakdown 5 Min (Phase 1)

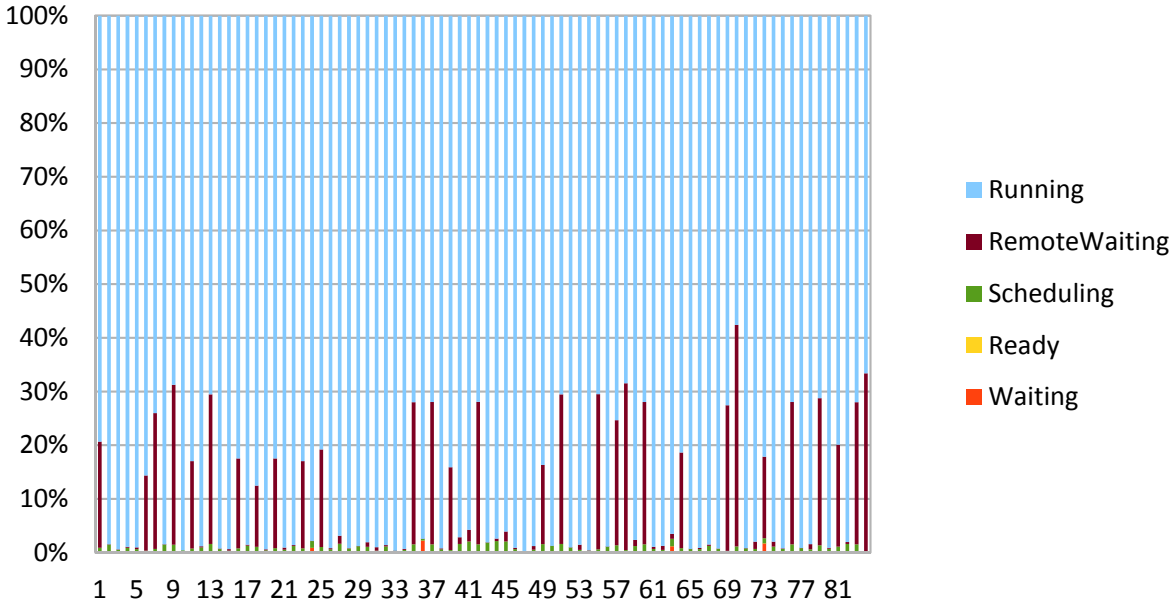


Figure 4.6: Execution Time Breakdown 2.5 Min (Phase 1)

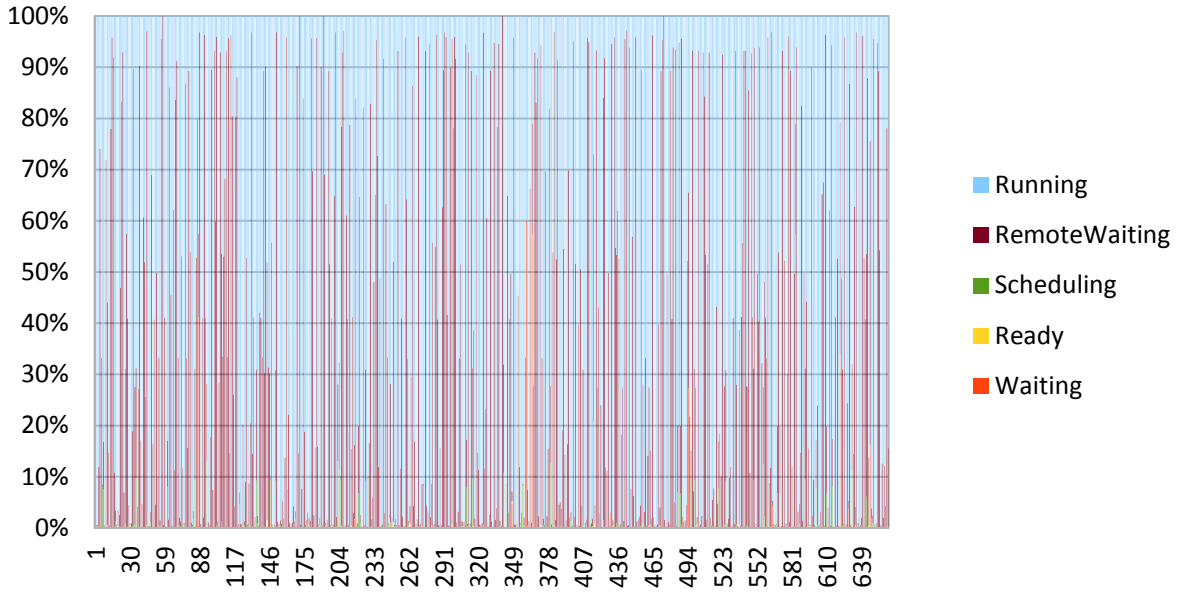


Figure 4.7: Execution Time Breakdown 10 Min (Phase 2)

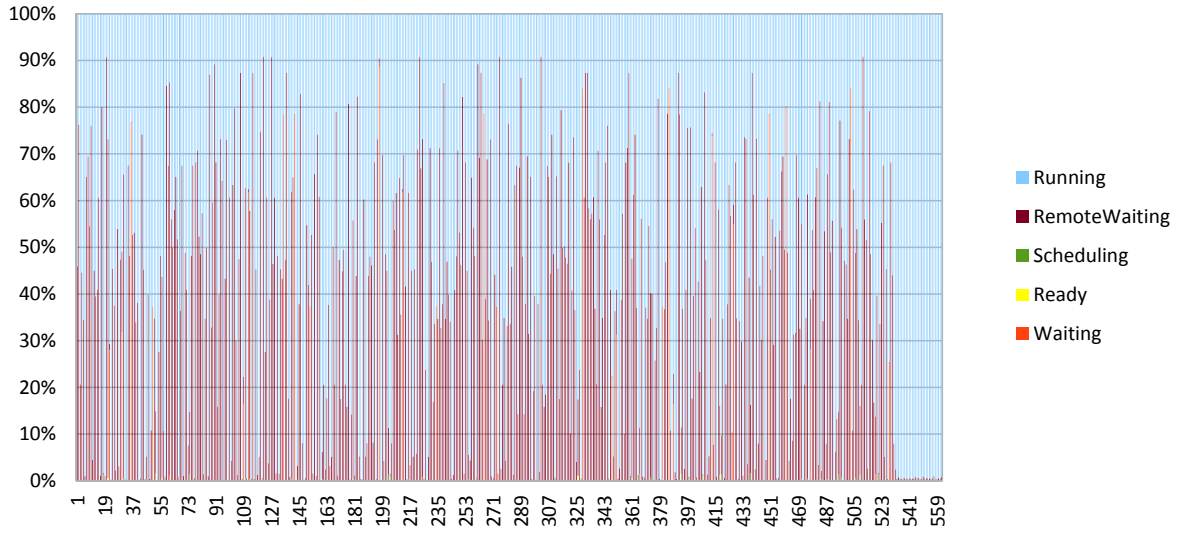


Figure 4.8: Execution Time Breakdown 5 Min (Phase 2)

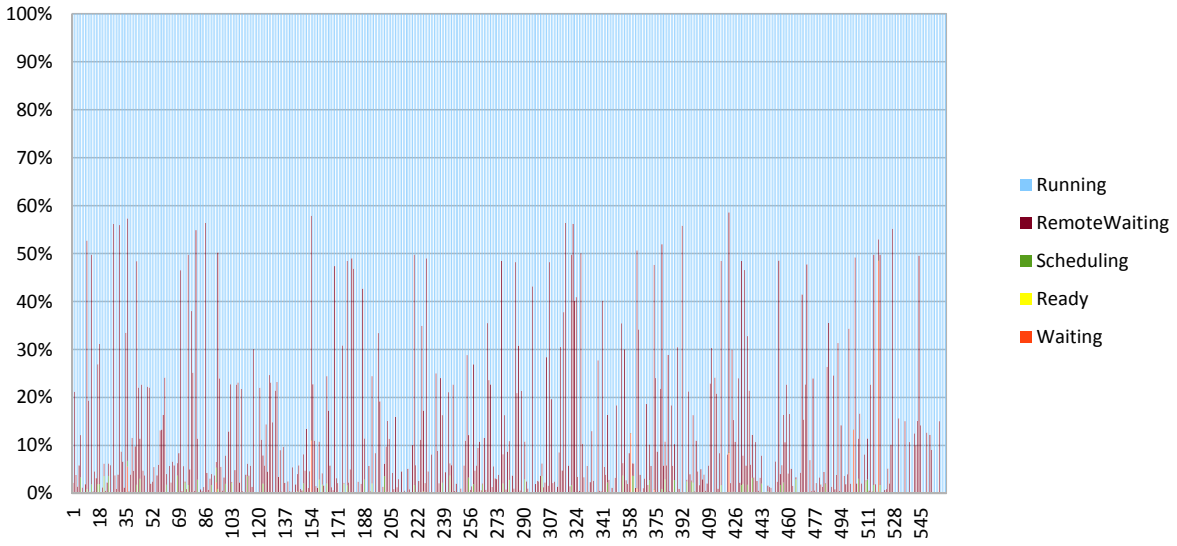


Figure 4.9: Execution Time Breakdown 2.5 Min (Phase 2)

For example machine 4 in Figure 4.10 succeeded to run all 10 jobs and it has very low total execution time for all jobs compared to other machines. Similarly machines 43 and 59 have their smallest total execution time but the fact is that machine 43 never succeeded to execute even a single job of batch of 10 jobs. All the jobs aborted in this machine. While machine 59 successfully completed the execution of all 10 jobs of 10 Min data-set resolution. The last machine in Figure 4.10 successfully executed 2 jobs out of 10.

4.3.2 Failed or Aborted jobs

Figures: 4.13, 4.14 and 4.15 show the total number of jobs failed or aborted per machine in phase 2 for data-set 10 Min, 5 Min and 2.5 Min respectively. For example in Figure 4.14, the number of aborted jobs on machine 11 is 10. This machine didn't manage to execute any job.

While there is only one job aborted and 9 successfully terminated on machine 83 in Figure 4.14.

There might be many reasons involved in this failure or abortion of jobs. The jobs that need to fetch larger files have very high failure rate which means independent of the environment we should avoid as much as possible to transfer large files (larger

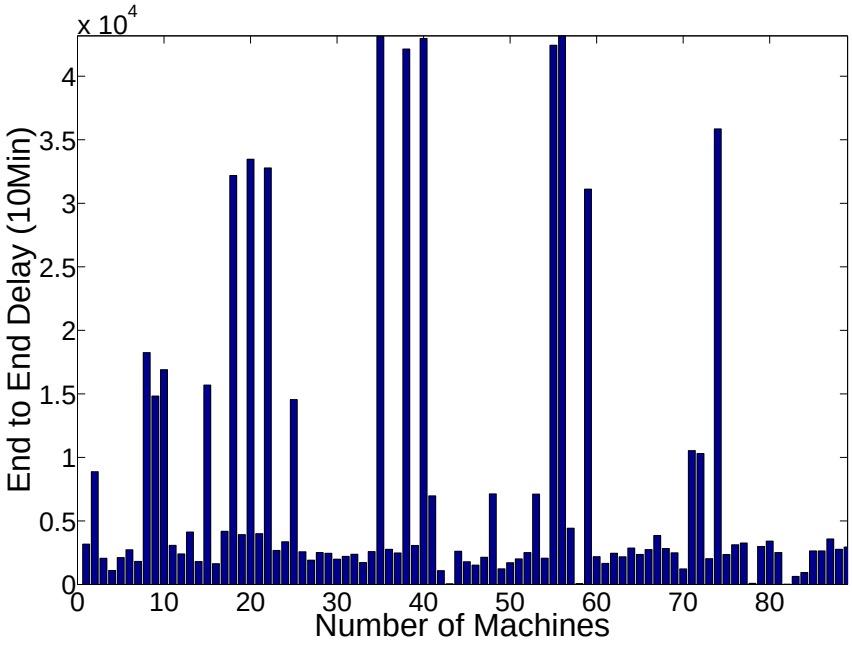


Figure 4.10: End to End Delay of Jobs (10 Min Data-set)

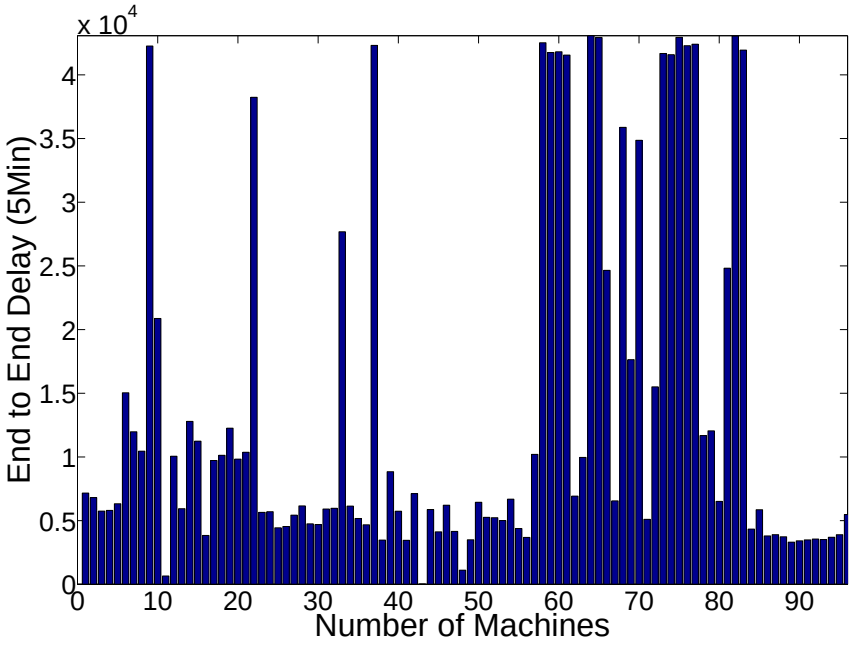


Figure 4.11: End to End Delay of Jobs (5m Data-set)

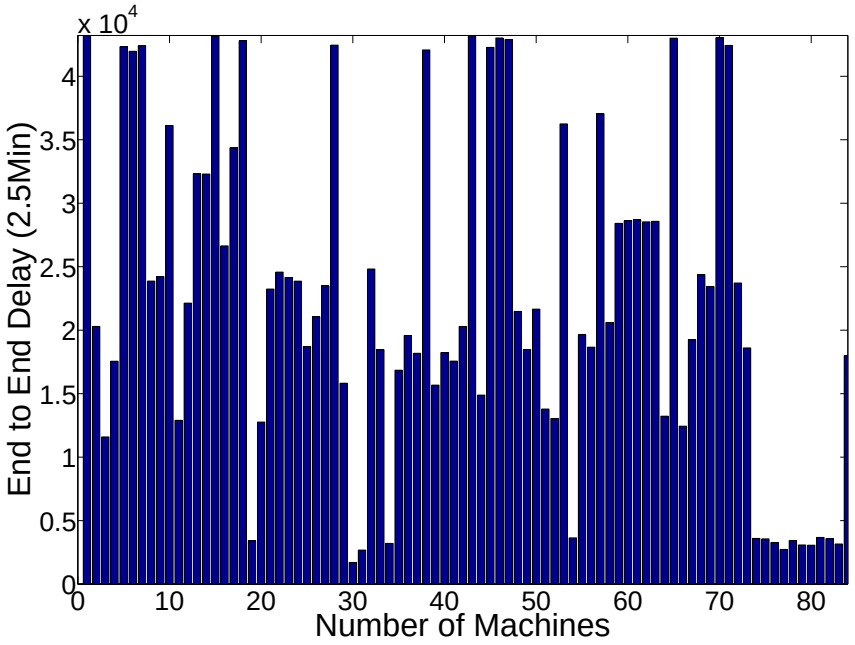


Figure 4.12: End to End Delay of Jobs (2.5m Data-set)

than few megabytes) across the network to do any kind of processing. Some of the jobs had exit code different of zero or exit code equals to zero but were not executed because of various "logged reasons". Many jobs could not continue their execution and were aborted because they had proxy expired. The cancellation of the jobs by CE admin is also one of the jobs failure reason. Lack of disk space also matters in the successful execution of jobs. Therefore, making an initial good selection of machines is very important.

Figures 4.16, 4.17 and 4.18 represent the aborted and the successful jobs of the corresponding data-set. The white area in the graphs shows the successful jobs while the blue area represents the failed or aborted jobs of the specific data-set.

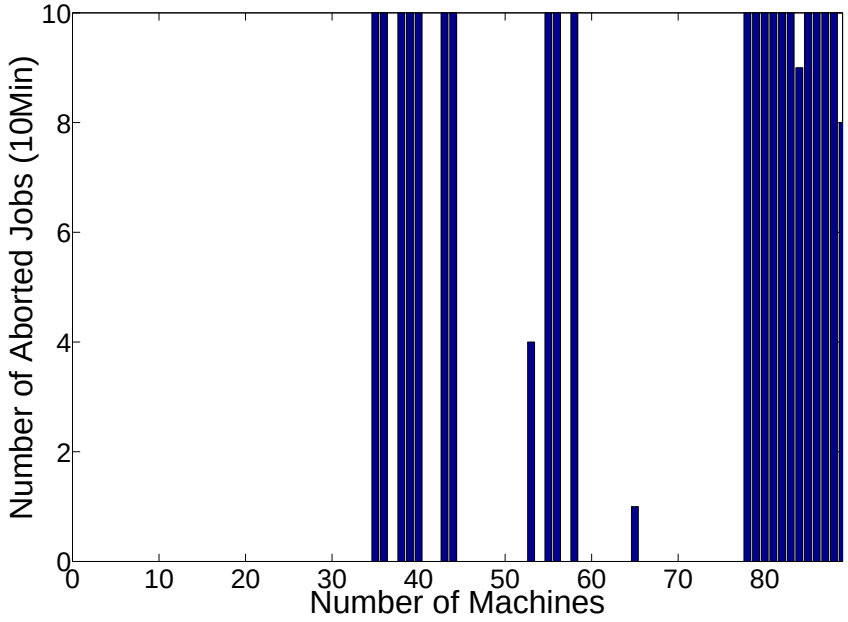


Figure 4.13: Number of Jobs Aborted per Machine (10 Min Data-set)

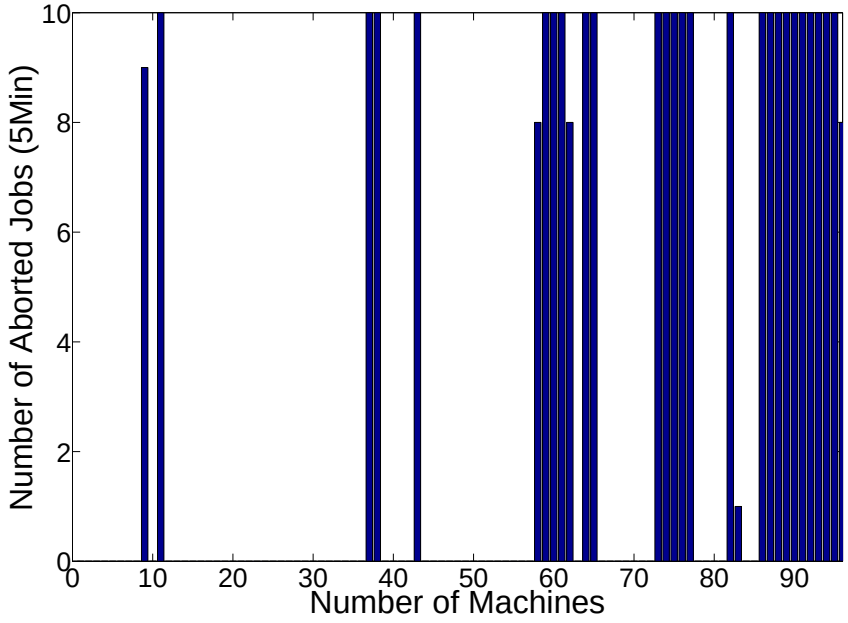


Figure 4.14: Number of Jobs aborted per Machine (5m Data-set)

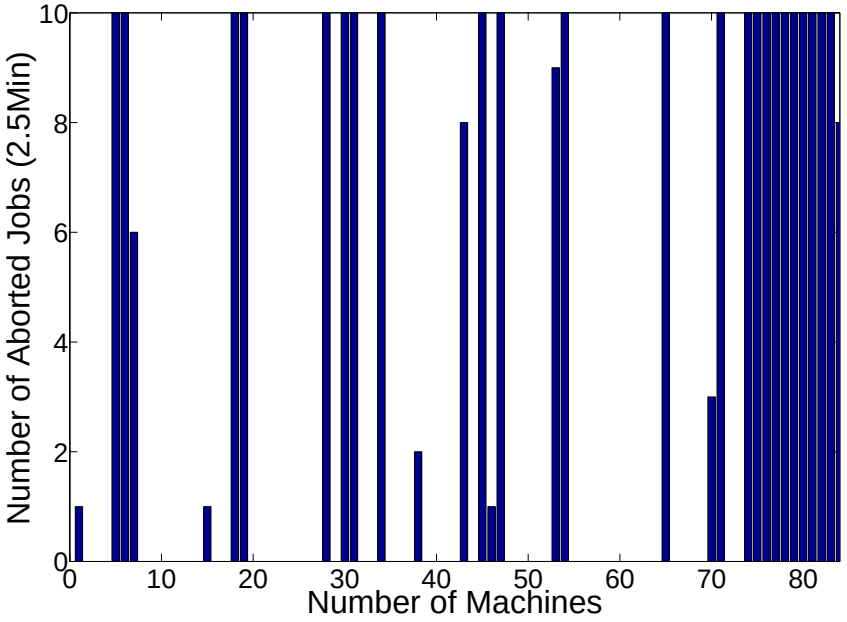


Figure 4.15: Number of Jobs Aborted per Machine (2.5m Data-set)

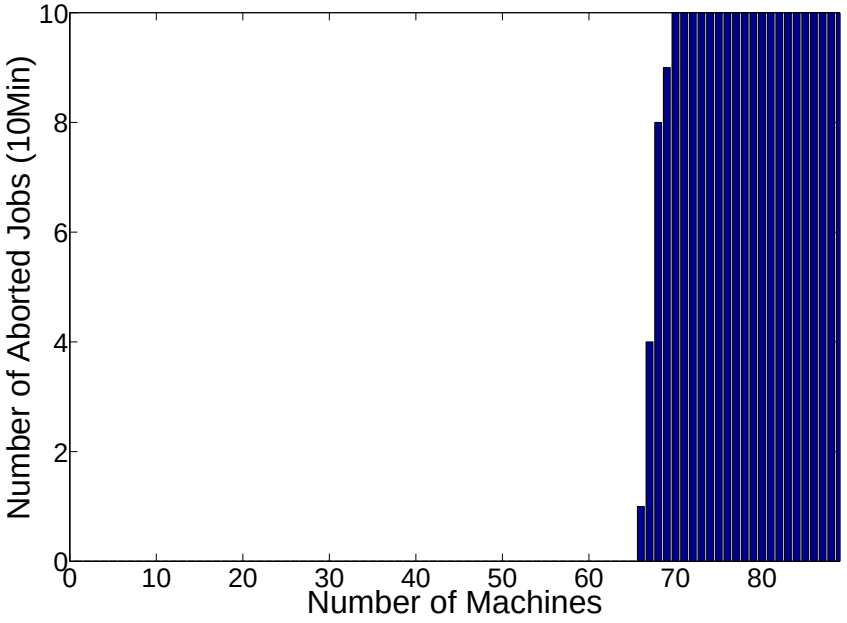


Figure 4.16: Total Number of Aborted and Successful Jobs (10 Min Data-set)

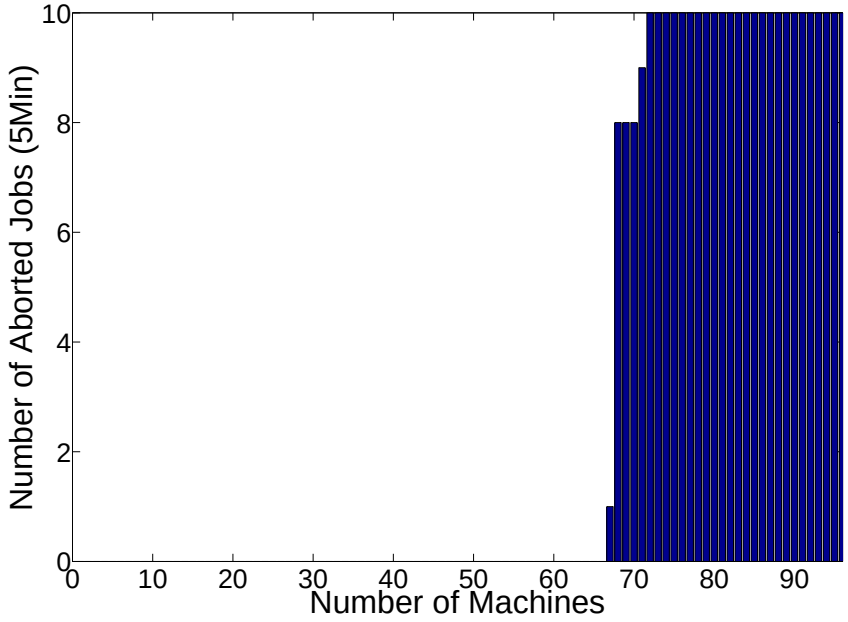


Figure 4.17: Total Number Aborted and Successful Jobs (5 Min Data-set)

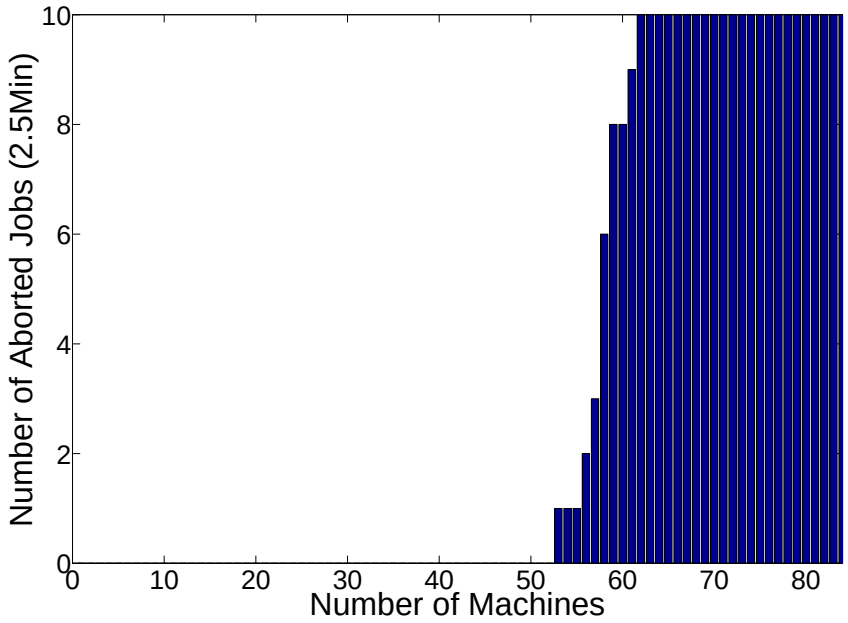


Figure 4.18: Total Number of Aborted and Sucessful Jobs (2.5 Min Data-set)

Chapter 5

Conclusions and Future Work

Grid infrastructures offer a good opportunity for harnessing resources of all types, which are made available through coordinated sharing policies. Grid middleware allows researchers to access these resources through Virtual Organizations in order to advance their science using the resources available to make experiments. However, despite the success of utilization of this infrastructure by researchers, there is still scope for improvements. First, there are still several communities that could benefit from using those resources, but do not know how. Second, various reasons prevent more people from using grid infrastructures: (1) the bureaucracy involved in using grids (for example, obtaining certificates or the need to have an account on a user interface), (2) the overheads of grid services due to the various layers of software and centralized servers, (3) the high rate of job failures, and (4) the change in methodology and environment to execute applications.

In this work, we show that totally relying on a grid middleware, like gLite, to execute jobs that require data transfers can be very inefficient. The successful execution of these jobs depends on a good choice of machines. Specially when applications need to transfer and process remote data. We have shown that a careful selection of resources can reduce the failure rate in 30%. Based on these results, we proposed a methodology to select machines when starting a grid application according to the size of the data transfers. We also proposed various scheduling strategies that can be dynamically adapted during execution using the initial rank of machines that were selected to start the application.

One of the issues not addressed in this work is the size limit of the user's sandbox.

Our methodology and strategies assume that the data can fit in the sandbox. One interesting path to follow would be to use the same methodology to rank machines based on data stored on the Storage Elements. Actually, in this work, we fetch data from a central server. We could use the Storage Element to store several of the datasets with different resolutions and guide the choice for a machine according to the distance to the data source chosen, as other schedulers, including gLite already do.

Appendix A

Tables generated in Phases 1 and 2

No	Machines In the Grid	10 Min	5 Min	2.5 Min
1	arc-ce01.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
2	arc-ce02.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
3	arc-ce03.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
4	cale.uniandes.edu.co:8443/cream-pbs-biomed	Successful	Successful	Failed
5	cccreamceli09.in2p3.fr:8443/cream-sge-long	Failed	Failed	Failed
6	cccreamceli09.in2p3.fr:8443/cream-sge-medium	Failed	Failed	Failed
7	cccreamceli09.in2p3.fr:8443/cream-sge-short	Failed	Failed	Failed
8	cccreamceli10.in2p3.fr:8443/cream-sge-long	Failed	Failed	Failed
9	cccreamceli10.in2p3.fr:8443/cream-sge-medium	Failed	Failed	Failed
10	cccreamceli10.in2p3.fr:8443/cream-sge-short	Failed	Failed	Failed
11	cccreamceli11.in2p3.fr:8443/cream-sge-long	Failed	Failed	Failed
12	cccreamceli11.in2p3.fr:8443/cream-sge-medium	Failed	Failed	Failed
13	cccreamceli11.in2p3.fr:8443/cream-sge-short	Failed	Failed	Failed
14	cce.ihep.ac.cn:8443/cream-pbs-biomed	Failed	Failed	Failed
15	ce-01.roma3.infn.it:8443/cream-pbs-grid	Successful	Successful	Successful
16	ce-02.roma3.infn.it:8443/cream-pbs-grid	Failed	Successful	Failed
17	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Successful	Successful	Successful
18	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Successful	Successful	Successful
19	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Successful	Successful	Failed
20	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Successful	Successful	Successful
21	ce.scope.unina.it:8443/cream-pbs-egee_long	Successful	Failed	Failed
22	ce.scope.unina.it:8443/cream-pbs-egee_short	Successful	Successful	Successful
23	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Successful	Successful	Failed
24	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Failed	Successful	Failed
25	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-sunx4600	Failed	Successful	Successful
26	ce.ulakbim.gov.tr:8443/cream-pbs-biomed	Failed	Failed	Failed
27	ce0.bordeaux.inra.fr:8443/cream-pbs-biomed	Successful	Successful	Successful

No	Machines In the Grid	10 Min	5 Min	2.5 Min
28	ce0.bordeaux.inra.fr:8443/cream-pbs-sdj	Failed	Successful	Failed
29	ce0.m3pec.u-bordeaux1.fr:8443/cream-pbs-biomed	Successful	Successful	Successful
30	ce0.m3pec.u-bordeaux1.fr:8443/cream-pbs-sdj	Failed	Failed	Failed
31	ce01-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Successful
32	ce01.gridc.lip.pt:8443/cream-sge-gridq	Failed	Failed	Failed
33	ce01.tier2.hep.manchester.ac.uk:8443/cream-pbs-long	Failed	Failed	Failed
34	ce01.up.pt:8443/cream-pbs-biomed	Successful	Failed	Failed
35	ce02.lip.pt:8443/cream-sge-gridq	Failed	Failed	Failed
36	ce02.ngcc.acad.bg:8443/cream-pbs-biomed	Successful	Failed	Successful
37	ce02.tier2.hep.manchester.ac.uk:8443/cream-pbs-long	Successful	Successful	Successful
38	ce02.up.pt:8443/cream-pbs-biomed	Successful	Failed	Failed
39	ce04-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Successful
40	ce05-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Successful
41	ce05.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Successful	Failed	Failed
42	ce05.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_2G_long	Failed	Failed	Failed
43	ce06-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Successful
44	ce06.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Successful	Failed	Failed
45	ce06.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_2G_long	Failed	Failed	Failed
46	ce06.ncg.ingrid.pt:8443/cream-sge-gridq	Failed	Failed	Failed
47	ce07-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Failed
48	ce07.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Successful	Failed	Failed
49	ce07.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_2G_long	Failed	Failed	Failed
50	ce08-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Successful	Successful	Successful
51	ce1.ts.infn.it:8443/cream-lsf-grid	Successful	Successful	Successful
52	ce101.grid.ucy.ac.cy:8443/cream-pbs-biomed	Successful	Successful	Successful
53	ce3.ppgrid1.rhul.ac.uk:8443/cream-pbs-biomed	Failed	Successful	Failed
54	ce3.ui.savba.sk:8443/cream-pbs-biomed	Successful	Successful	Successful
55	ceprod05.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Successful	Successful	Successful
56	ceprod06.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Successful	Successful	Successful
57	ceprod07.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Successful	Successful	Successful
58	ceprod08.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Successful	Successful	Successful
59	cert-37.pd.infn.it:8443/cream-lsf-grid	Successful	Successful	Successful
60	cirigridce01.univ-bpclermont.fr:8443/cream-pbs-biomed	Successful	Successful	Successful
61	clrccece01.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
62	clrccece02.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
63	clrccece03.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
64	cox01.grid.metu.edu.tr:8443/cream-pbs-biomed	Successful	Successful	Successful
65	cr1.ipp.acad.bg:8443/cream-pbs-biomed	Successful	Successful	Failed
66	cream-ce-2.ba.infn.it:8443/cream-pbs-infinite	Failed	Failed	Successful
67	cream-ce-2.ba.infn.it:8443/cream-pbs-long	Failed	Failed	Successful
68	cream-ce-2.ba.infn.it:8443/cream-pbs-short	Failed	Failed	Successful
69	cream-ce-3.ba.infn.it:8443/cream-pbs-infinite	Failed	Failed	Successful
70	cream-ce-3.ba.infn.it:8443/cream-pbs-long	Failed	Failed	Successful

No	Machines In the Grid	10 Min	5 Min	2.5 Min
71	cream-ce-3.ba.infn.it:8443/cream-pbs-short	Failed	Failed	Successful
72	cream-ce-grid.obspm.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
73	cream-ce.cat.cbpf.br:8443/cream-pbs-biomed	Failed	Successful	Failed
74	cream-ce01.ariagni.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Failed	Successful
75	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Successful	Failed	Successful
76	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Successful	Successful	Successful
77	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Successful	Successful	Successful
78	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Successful	Successful
79	cream-ce02.cat.cbpf.br:8443/cream-pbs-biomed	Failed	Successful	Failed
80	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Successful	Successful	Successful
81	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Successful	Successful	Successful
82	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Successful	Successful	Successful
83	cream-ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Failed	Failed
84	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Successful	Successful
85	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Successful	Successful	Successful
86	cream.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Failed	Successful	Failed
87	cream.grid.uni-sofia.bg:8443/cream-pbs-biomed	Successful	Successful	Successful
88	cream01-tic.ciemat.es:8443/cream-pbs-biomed	Successful	Successful	Failed
89	cream01.grid.auth.gr:8443/cream-pbs-biomed	Successful	Successful	Successful
90	cream01.grid.uoi.gr:8443/cream-pbs-biomed	Successful	Successful	Failed
91	cream01.kallisto.hellasgrid.gr:8443/cream-pbs-biomed	Successful	Successful	Successful
92	cream02.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Failed	Successful	Failed
93	cream2.ppgrid1.rhul.ac.uk:8443/cream-pbs-biomed	Failed	Successful	Successful
94	creamce.gina.sara.nl:8443/cream-pbs-medium	Failed	Failed	Failed
95	creamce.gina.sara.nl:8443/cream-pbs-short	Failed	Failed	Failed
96	creamce.reef.man.poznan.pl:8443/cream-pbs-biomed	Successful	Failed	Failed
97	creamce02.ciemat.es:8443/cream-pbs-medium	Successful	Successful	Successful
98	creamce03.ciemat.es:8443/cream-pbs-medium	Successful	Successful	Successful
99	creamce2.gina.sara.nl:8443/cream-pbs-medium	Failed	Failed	Failed
100	creamce2.gina.sara.nl:8443/cream-pbs-short	Failed	Failed	Failed
101	creamce3.gina.sara.nl:8443/cream-pbs-medium	Failed	Failed	Failed
102	creamce3.gina.sara.nl:8443/cream-pbs-short	Failed	Failed	Failed
103	cygnus.grid.rug.nl:8443/cream-pbs-medium	Successful	Successful	Successful
104	cygnus.grid.rug.nl:8443/cream-pbs-short	Successful	Failed	Successful
105	dc2-grid-66.brunel.ac.uk:8443/cream-pbs-biomed	Failed	Successful	Successful
106	dc2-grid-68.brunel.ac.uk:8443/cream-pbs-biomed	Failed	Failed	Failed
107	dc2-grid-70.brunel.ac.uk:8443/cream-pbs-biomed	Failed	Failed	Successful
108	desdemona.zih.tu-dresden.de:8443/cream-pbs-gridexpr_scli	Failed	Failed	Failed
109	desdemona.zih.tu-dresden.de:8443/cream-pbs-gridlong_scli	Failed	Successful	Failed
110	desdemona.zih.tu-dresden.de:8443/cream-pbs-gridmedium_scli	Failed	Failed	Failed
111	desdemona.zih.tu-dresden.de:8443/cream-pbs-gridshort_scli	Failed	Failed	Failed
112	desdemona.zih.tu-dresden.de:8443/cream-pbs-route_scli	Failed	Failed	Failed
113	dissel.nikhef.nl:2119/jobmanager-pbs-gratis	Failed	Failed	Failed

No	Machines In the Grid	10 Min	5 Min	2.5 Min
114	dwarf.wcss.wroc.pl:8443/cream-pbs-biomed	Successful	Successful	Successful
115	epgr02.ph.bham.ac.uk:8443/cream-pbs-long	Failed	Successful	Failed
116	epgr02.ph.bham.ac.uk:8443/cream-pbs-short	Failed	Successful	Failed
117	fal-pygrid-44.lancs.ac.uk:8443/cream-pbs-q	Failed	Failed	Failed
118	fornax-ce.itwm.fhg.de:8443/cream-pbs-biomed	Failed	Failed	Failed
119	fornax-ce2.itwm.fhg.de:8443/cream-pbs-biomed	Failed	Failed	Failed
120	gazon.nikhef.nl:8443/cream-pbs-gratis	Failed	Failed	Failed
121	glite-cream.scai.fraunhofer.de:8443/cream-pbs-egbiomed	Successful	Successful	Failed
122	grid-cr0.desy.de:8443/cream-pbs-desy	Successful	Successful	Successful
123	grid-cr1.desy.de:8443/cream-pbs-desy	Successful	Failed	Successful
124	grid-cr2.desy.de:8443/cream-pbs-desy	Failed	Successful	Successful
125	grid-cr3.desy.de:8443/cream-pbs-desy	Successful	Successful	Successful
126	grid-cr4.desy.de:8443/cream-pbs-desy	Successful	Successful	Successful
127	grid0.fe.infn.it:8443/cream-pbs-grid	Successful	Successful	Successful
128	grid001.fc.up.pt:8443/cream-pbs-biomed	Successful	Failed	Failed
129	grid001.fe.up.pt:8443/cream-pbs-biomed	Successful	Failed	Failed
130	grid001.ics.forth.gr:8443/cream-pbs-biomed	Failed	Failed	Failed
131	grid002.jet.efda.org:8443/cream-pbs-biomed	Successful	Failed	Successful
132	grid36.lal.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
133	grid36.lal.in2p3.fr:8443/cream-pbs-sdj	Failed	Failed	Failed
134	gridce.ilc.cnr.it:8443/cream-pbs-grid	Failed	Failed	Failed
135	gridce0.pi.infn.it:8443/cream-lsf-biomed	Failed	Not Available	Failed
136	gridce01.ifca.es:8443/cream-sge-biomed	Failed	Successful	Failed
137	gridce02.ifca.es:8443/cream-sge-biomed	Failed	Failed	Failed
138	gridce03.ifca.es:8443/cream-sge-biomed	Failed	Successful	Failed
139	gridce1.pi.infn.it:8443/cream-lsf-biomed	Failed	Not Available	Failed
140	gridce2.pi.infn.it:8443/cream-lsf-biomed	Failed	Not Available	Failed
141	gridsrv2-4.dir.garr.it:8443/cream-pbs-grid	Successful	Successful	Successful
142	grisuce.scope.unina.it:8443/cream-pbs-grisu_long	Successful	Failed	Failed
143	grisuce.scope.unina.it:8443/cream-pbs-grisu_short	Successful	Successful	Successful
144	hepgrid10.ph.liv.ac.uk:8443/cream-pbs-long	Failed	Successful	Failed
145	hepgrid5.ph.liv.ac.uk:8443/cream-pbs-long	Failed	Successful	Failed
146	hepgrid6.ph.liv.ac.uk:8443/cream-pbs-long	Failed	Successful	Failed
147	hepgrid97.ph.liv.ac.uk:8443/cream-pbs-long	Failed	Successful	Failed
148	heplnx206.pp.rl.ac.uk:8443/cream-pbs-grid	Failed	Failed	Failed
149	heplnx207.pp.rl.ac.uk:8443/cream-pbs-grid	Failed	Failed	Failed
150	heplnx208.pp.rl.ac.uk:8443/cream-pbs-grid	Failed	Failed	Failed
151	juk.nikhef.nl:8443/cream-pbs-gratis	Failed	Failed	Failed
152	kalkan1.ulakbim.gov.tr:8443/cream-pbs-biomed	Successful	Successful	Successful
153	klomp.nikhef.nl:8443/cream-pbs-gratis	Failed	Failed	Failed
154	lcgce12.jinr.ru:8443/cream-pbs-biomed	Successful	Successful	Successful
155	lcgce2.shef.ac.uk:8443/cream-pbs-biomed	Failed	Failed	Failed
156	lcgce21.jinr.ru:8443/cream-pbs-biomed	Successful	Successful	Successful

No	Machines In the Grid	10 Min	5 Min	2.5 Min
157	lcgce3.shef.ac.uk:8443/cream-pbs-biomed	Failed	Failed	Failed
158	linux1.grid.creatis.insa-lyon.fr:8443/cream-pbs-qbiomed	Successful	Successful	Successful
159	llrcream.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
160	llrcream.in2p3.fr:8443/cream-pbs-sdj	Failed	Failed	Failed
161	lpsc-ce.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
162	lpsc-ce2.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
163	lpsc-cream-ce.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
164	lptace01.msfg.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
165	lptace01.msfg.fr:8443/cream-pbs-sdj	Failed	Failed	Failed
166	marcream01.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
167	marcream02.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
168	ngiescream.i3m.upv.es:8443/cream-pbs-biomed	Successful	Successful	Successful
169	node01-04.grid.renam.md:8443/cream-pbs-other	Failed	Successful	Failed
170	node05-02.imi.renam.md:8443/cream-pbs-other	Successful	Successful	Successful
171	node74.datagrid.cea.fr:8443/cream-pbs-biomed	Successful	Successful	Successful
172	prod-ce-01.pd.infn.it:8443/cream-lsf-grid	Successful	Successful	Successful
173	sampace.if.usp.br:8443/cream-pbs-biomed	Failed	Failed	Failed
174	sbgce2.in2p3.fr:8443/cream-pbs-biomed	Failed	Failed	Failed
175	snf-10952.vm.okeanos.grnet.gr:8443/cream-pbs-biomed	Successful	Failed	Failed
176	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Successful	Successful	Successful
177	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
178	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Successful	Successful	Successful
179	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
180	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Successful	Successful	Successful
181	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
182	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Successful	Successful	Successful
183	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
184	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Successful	Successful	Successful
185	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Successful	Successful	Successful
186	t2-ce-01.to.infn.it:8443/cream-pbs-biomed	Failed	Failed	Failed
187	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Successful	Successful	Successful
188	wario.univ-lille1.fr:8443/cream-pbs-biomed	Failed	Failed	Failed

Table A.1: Successful and Failed Machines while Running the One Job of each Data Set

No	Machines In the Grid	10 Minute
1	cale.uniandes.edu.co:8443/cream-pbs-biomed	Good
2	ce-01.roma3.infn.it:8443/cream-pbs-grid	Good
3	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
4	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Good
5	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
6	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
7	ce.scope.unina.it:8443/cream-pbs-egee_long	Good
8	ce.scope.unina.it:8443/cream-pbs-egee_short	Good
9	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
10	ce0.bordeaux.inra.fr:8443/cream-pbs-biomed	Good
11	ce0.m3pec.u-bordeaux1.fr:8443/cream-pbs-biomed	Good
12	ce01-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
13	ce01.up.pt:8443/cream-pbs-biomed	Good
14	ce02.ngcc.acad.bg:8443/cream-pbs-biomed	Good
15	ce02.tier2.hep.manchester.ac.uk:8443/cream-pbs-long	Good
16	ce02.up.pt:8443/cream-pbs-biomed	Good
17	ce04-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
18	ce05-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
19	ce05.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Good
20	ce06-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
21	ce06.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Good
22	ce07-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
23	ce07.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long	Good
24	ce08-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
25	ce1.ts.infn.it:8443/cream-lsf-grid	Good
26	ce101.grid.ucy.ac.cy:8443/cream-pbs-biomed	Good
27	ce3.ui.savba.sk:8443/cream-pbs-biomed	Good
28	ceprod05.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
29	ceprod06.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
30	ceprod07.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
31	ceprod08.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
32	cert-37.pd.infn.it:8443/cream-lsf-grid	Good
33	cirigridce01.univ-bpclermont.fr:8443/cream-pbs-biomed	Good
34	cox01.grid.metu.edu.tr:8443/cream-pbs-biomed	Good
35	cr1.ipp.acad.bg:8443/cream-pbs-biomed	Bad
36	cream-ce01.ariagni.hellasgrid.gr:8443/cream-pbs-biomed	Bad
37	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
38	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
39	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
40	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Good
41	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
42	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
43	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good

No	Machines In the Grid	10 Minute
44	cream-ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed	Good
45	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Bad
46	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Good
47	cream.grid.uni-sofia.bg:8443/cream-pbs-biomed	Bad
48	cream01-tic.ciemat.es:8443/cream-pbs-biomed	Good
49	cream01.grid.auth.gr:8443/cream-pbs-biomed	Bad
50	cream01.grid.uoi.gr:8443/cream-pbs-biomed	Bad
51	cream01.kallisto.hellasgrid.gr:8443/cream-pbs-biomed	Good
52	creamce.reef.man.poznan.pl:8443/cream-pbs-biomed	Bad
53	creamce02.ciemat.es:8443/cream-pbs-medium	Good
54	creamce03.ciemat.es:8443/cream-pbs-medium	Good
55	cygnus.grid.rug.nl:8443/cream-pbs-medium	Bad
56	cygnus.grid.rug.nl:8443/cream-pbs-short	Bad
57	dwarf.wcss.wroc.pl:8443/cream-pbs-biomed	Good
58	glite-cream.scai.fraunhofer.de:8443/cream-pbs-egbiomed	Bad
59	grid-cr0.desy.de:8443/cream-pbs-desy	Good
60	grid-cr1.desy.de:8443/cream-pbs-desy	Good
61	grid-cr3.desy.de:8443/cream-pbs-desy	Good
62	grid-cr4.desy.de:8443/cream-pbs-desy	Good
63	grid0.fe.infn.it:8443/cream-pbs-grid	Good
64	grid001.fc.up.pt:8443/cream-pbs-biomed	Good
65	grid001.fe.up.pt:8443/cream-pbs-biomed	Good
66	grid002.jet.efda.org:8443/cream-pbs-biomed	Good
67	gridsrv2-4.dir.garr.it:8443/cream-pbs-grid	Good
68	grisuce.scope.unina.it:8443/cream-pbs-grisu_long	Good
69	grisuce.scope.unina.it:8443/cream-pbs-grisu_short	Good
70	kalkan1.ulakbim.gov.tr:8443/cream-pbs-biomed	Good
71	lcgce12.jinr.ru:8443/cream-pbs-biomed	Good
72	lcgce21.jinr.ru:8443/cream-pbs-biomed	Good
73	linux1.grid.creatis.insa-lyon.fr:8443/cream-pbs-qbiomed	Good
74	ngiescream.i3m.upv.es:8443/cream-pbs-biomed	Good
75	node05-02.imi.renam.md:8443/cream-pbs-other	Good
76	node74.datagrid.cea.fr:8443/cream-pbs-biomed	Good
77	prod-ce-01.pd.infn.it:8443/cream-lsf-grid	Good
78	snf-10952.vm.okeanos.grnet.gr:8443/cream-pbs-biomed	Bad
79	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
80	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
81	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
82	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
83	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
84	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
85	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
86	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad

No	Machines In the Grid	10 Minute
87	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
88	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
89	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Good

Table A.2: Good and Bad Machines for 10 Minute Data Set

No	Machines In the Grid	5 Minute
1	cale.uniandes.edu.co:8443/cream-pbs-biomed	Good
2	ce-01.roma3.infn.it:8443/cream-pbs-grid	Good
3	ce-02.roma3.infn.it:8443/cream-pbs-grid	Good
4	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
5	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Good
6	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
7	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
8	ce.scope.unina.it:8443/cream-pbs-egee_short	Good
9	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
10	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
11	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-sunx4600	Bad
12	ce0.bordeaux.inra.fr:8443/cream-pbs-biomed	Good
13	ce0.bordeaux.inra.fr:8443/cream-pbs-sdj	Bad
14	ce0.m3pec.u-bordeaux1.fr:8443/cream-pbs-biomed	Good
15	ce01-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
16	ce02.tier2.hep.manchester.ac.uk:8443/cream-pbs-long	Good
17	ce04-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
18	ce05-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
19	ce06-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
20	ce07-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
21	ce08-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
22	ce1.ts.infn.it:8443/cream-lsf-grid	Good
23	ce101.grid.ucy.ac.cy:8443/cream-pbs-biomed	Good
24	ce3.ppgrid1.rhul.ac.uk:8443/cream-pbs-biomed	Good
25	ce3.ui.savba.sk:8443/cream-pbs-biomed	Good
26	ceprod05.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
27	ceprod06.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
28	ceprod07.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
29	ceprod08.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
30	cert-37.pd.infn.it:8443/cream-lsf-grid	Good
31	cirigridce01.univ-bpclermont.fr:8443/cream-pbs-biomed	Good
32	cox01.grid.metu.edu.tr:8443/cream-pbs-biomed	Good
33	cr1.ipp.acad.bg:8443/cream-pbs-biomed	Good
34	cream-ce.cat.cbpf.br:8443/cream-pbs-biomed	Good
35	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
36	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
37	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Bad
38	cream-ce02.cat.cbpf.br:8443/cream-pbs-biomed	Good
39	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
40	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
41	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
42	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Good
43	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Good

No	Machines In the Grid	5 Minute
44	cream.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Good
45	cream.grid.uni-sofia.bg:8443/cream-pbs-biomed	Bad
46	cream01-tic.ciemat.es:8443/cream-pbs-biomed	Good
47	cream01.grid.auth.gr:8443/cream-pbs-biomed	Bad
48	cream01.grid.uoi.gr:8443/cream-pbs-biomed	Good
49	cream01.kallisto.hellasgrid.gr:8443/cream-pbs-biomed	Good
50	cream02.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Good
51	cream2.ppgrid1.rhul.ac.uk:8443/cream-pbs-biomed	Bad
52	creamce02.ciemat.es:8443/cream-pbs-medium	Good
53	creamce03.ciemat.es:8443/cream-pbs-medium	Good
54	cygnus.grid.rug.nl:8443/cream-pbs-medium	Good
55	dc2-grid-66.brunel.ac.uk:8443/cream-pbs-biomed	Good
56	desdemona.zih.tu-dresden.de:8443/cream-pbs-gridlong_scli	Good
57	dwarf.wcss.wroc.pl:8443/cream-pbs-biomed	Bad
58	epgr02.ph.bham.ac.uk:8443/cream-pbs-long	Bad
59	epgr02.ph.bham.ac.uk:8443/cream-pbs-short	Bad
60	glite-cream.scai.fraunhofer.de:8443/cream-pbs-egbiomed	Good
61	grid-cr0.desy.de:8443/cream-pbs-desy	Good
62	grid-cr2.desy.de:8443/cream-pbs-desy	Good
63	grid-cr3.desy.de:8443/cream-pbs-desy	Good
64	grid-cr4.desy.de:8443/cream-pbs-desy	Good
65	grid0.fe.infn.it:8443/cream-pbs-grid	Good
66	gridce01.ifca.es:8443/cream-sge-biomed	Bad
67	gridce03.ifca.es:8443/cream-sge-biomed	Bad
68	gridsrv2-4.dir.garr.it:8443/cream-pbs-grid	Good
69	grisuce.scope.unina.it:8443/cream-pbs-grisu_short	Good
70	hepgrid10.ph.liv.ac.uk:8443/cream-pbs-long	Bad
71	hepgrid5.ph.liv.ac.uk:8443/cream-pbs-long	Bad
72	hepgrid6.ph.liv.ac.uk:8443/cream-pbs-long	Bad
73	hepgrid97.ph.liv.ac.uk:8443/cream-pbs-long	Bad
74	kalkan1.ulakbim.gov.tr:8443/cream-pbs-biomed	Bad
75	lcgce12.jinr.ru:8443/cream-pbs-biomed	Good
76	lcgce21.jinr.ru:8443/cream-pbs-biomed	Good
77	linux1.grid.creatis.insa-lyon.fr:8443/cream-pbs-qbiomed	Good
78	ngiescream.i3m.upv.es:8443/cream-pbs-biomed	Good
79	node01-04.grid.renam.md:8443/cream-pbs-other	Bad
80	node05-02.imi.renam.md:8443/cream-pbs-other	Good
81	node74.datagrid.cea.fr:8443/cream-pbs-biomed	Good
82	prod-ce-01.pd.infn.it:8443/cream-lsf-grid	Good
83	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
84	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
85	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
86	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad

No	Machines In the Grid	5 Minute
87	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
88	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
89	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
90	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
91	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
92	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
93	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Good

Table A.3: Good and Bad Machines for 5 Minute Data Set

No	Machines In the Grid	2.5 Minute
1	ce-01.roma3.infn.it:8443/cream-pbs-grid	Good
2	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
3	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Good
4	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
5	ce.scope.unina.it:8443/cream-pbs-egee_short	Bad
6	ce.srce.egi.cro-ngi.hr:8443/cream-pbs-sunx4600	Bad
7	ce0.bordeaux.inra.fr:8443/cream-pbs-biomed	Good
8	ce0.m3pec.u-bordeaux1.fr:8443/cream-pbs-biomed	Good
9	ce01-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
10	ce02.ngcc.acad.bg:8443/cream-pbs-biomed	Good
11	ce02.tier2.hep.manchester.ac.uk:8443/cream-pbs-long	Good
12	ce04-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
13	ce05-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
14	ce06-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
15	ce08-lcg.cr.cnaf.infn.it:8443/cream-lsf-biomed	Good
16	ce1.ts.infn.it:8443/cream-lsf-grid	Bad
17	ce101.grid.ucy.ac.cy:8443/cream-pbs-biomed	Bad
18	ce3.ui.savba.sk:8443/cream-pbs-biomed	Good
19	ceprod05.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
20	ceprod06.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
21	ceprod07.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
22	ceprod08.grid.hep.ph.ic.ac.uk:8443/cream-sge-grid.q	Good
23	cert-37.pd.infn.it:8443/cream-lsf-grid	Good
24	cirigridce01.univ-bpclermont.fr:8443/cream-pbs-biomed	Good
25	cox01.grid.metu.edu.tr:8443/cream-pbs-biomed	Good
26	cream-ce-2.ba.infn.it:8443/cream-pbs-infinite	Bad
27	cream-ce-2.ba.infn.it:8443/cream-pbs-long	Good
28	cream-ce-2.ba.infn.it:8443/cream-pbs-short	Bad
29	cream-ce-3.ba.infn.it:8443/cream-pbs-infinite	Good
30	cream-ce-3.ba.infn.it:8443/cream-pbs-long	Good
31	cream-ce-3.ba.infn.it:8443/cream-pbs-short	Good
32	cream-ce01.ariagni.hellasgrid.gr:8443/cream-pbs-biomed	Bad
33	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
34	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
35	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
36	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Good
37	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
38	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
39	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
40	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Bad
41	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Good
42	cream.grid.uni-sofia.bg:8443/cream-pbs-biomed	Bad
43	cream01.grid.auth.gr:8443/cream-pbs-biomed	Bad

No	Machines In the Grid	2.5 Minute
44	cream01.kallisto.hellasgrid.gr:8443/cream-pbs-biomed	Good
45	cream2.ppgrid1.rhul.ac.uk:8443/cream-pbs-biomed	Good
46	creamce02.ciemat.es:8443/cream-pbs-medium	Good
47	creamce03.ciemat.es:8443/cream-pbs-medium	Good
48	cygnus.grid.rug.nl:8443/cream-pbs-medium	Good
49	cygnus.grid.rug.nl:8443/cream-pbs-short	Good
50	dc2-grid-66.brunel.ac.uk:8443/cream-pbs-biomed	Good
51	dc2-grid-70.brunel.ac.uk:8443/cream-pbs-biomed	Bad
52	dwarf.wcss.wroc.pl:8443/cream-pbs-biomed	Good
53	grid-cr0.desy.de:8443/cream-pbs-desy	Good
54	grid-cr1.desy.de:8443/cream-pbs-desy	Good
55	grid-cr2.desy.de:8443/cream-pbs-desy	Good
56	grid-cr3.desy.de:8443/cream-pbs-desy	Good
57	grid-cr4.desy.de:8443/cream-pbs-desy	Good
58	grid0.fe.infn.it:8443/cream-pbs-grid	Good
59	grid002.jet.efda.org:8443/cream-pbs-biomed	Good
60	gridsrv2-4.dir.garr.it:8443/cream-pbs-grid	Good
61	grisuce.scope.unina.it:8443/cream-pbs-grisu_short	Bad
62	kalkan1.ulakbim.gov.tr:8443/cream-pbs-biomed	Good
63	lcgce12.jinr.ru:8443/cream-pbs-biomed	Good
64	lcgce21.jinr.ru:8443/cream-pbs-biomed	Good
65	linux1.grid.creatis.insa-lyon.fr:8443/cream-pbs-qbiomed	Good
66	ngiescream.i3m.upv.es:8443/cream-pbs-biomed	Good
67	node05-02.imi.renam.md:8443/cream-pbs-other	Bad
68	node74.datagrid.cea.fr:8443/cream-pbs-biomed	Good
69	prod-ce-01.pd.infn.it:8443/cream-lsf-grid	Good
70	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
71	svr009.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
72	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
73	svr010.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
74	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
75	svr011.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
76	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
77	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
78	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
79	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
80	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Good

Table A.4: Good and Bad Machines for 2.5 Minute Data Set

References

[1] *Scientific Linux*. `https://www.scientificlinux.org/`.

[2] *Parallel Workload Archive*, July 2005. `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`.

[3] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with nimrod/g: killer application for the global grid? In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 520–528, 2000.

[4] The Globus Alliance. *The globus toolkit*. `http://www.globus.org/toolkit`.

[5] TF Ang, WK Ng, TC Ling, LY Por, and CS Liew. A bandwidth-aware job grouping-based scheduling on grid environment. *Information Technology Journal*, 8(3):372–377, 2009.

[6] F. Brasileiro, I. Dutra, and L. Ciuffo. *The EELA-2 e-Infrastructure User Guide*, July 2009. `http://documents.eu-eela.eu/record/1303/files/EELA-2_e-Infrastructure_User_Guide-v1.pdf?version=1`.

[7] Rajkumar Buyya, David Abramson, and Jonathan Giddy. An economy driven resource management architecture for global computational power grids, 2000.

[8] Min Cai, Shahram Ghandeharizadeh, Rolfe R. Schmidt, and Saihong Song. A comparison of alternative encoding mechanisms for web services. In Abdelkader Hameurlain, Rosine Cicchetti, and Roland Traunmller, editors, *DEXA*, volume 2453 of *Lecture Notes in Computer Science*, pages 93–102. Springer, 2002.

[9] David Carrera, Jordi Guitart, Vicenç Beltran, Jordi Torres, and Eduard Ayguadé. Performance impact of the grid middleware". Engineering the Grid: Status and Perspective, Section 5: Performance Aspects, Chapter 15, Jan 2006.

[10] Henri Casanova and Fran Berman. Parameter sweeps on the grid with apst. *Concurrency and Computation: Practice and Experience*, 1(1), 2002.

[11] Henri Casanova, Arnaud Legrand, Dmitrii Zagorodnov, and Francine Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *Heterogeneous Computing Workshop*, pages 349–363, 2000.

[12] Henri Casanova, Graziano Obertelli, Francine Berman, and Richard Wolski. The apples parameter sweep template: User-level middleware for the grid. In *Supercomputing, ACM/IEEE 2000 Conference*, pages 60–60, Nov 2000.

[13] K. Chiu, M. Govindaraju, and R. Bramley. Investigating the limits of SOAP performance for scientific computing. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 246–254, 2002.

[14] Konstantinos Christodoulopoulos, Vasileios Gkamas, and Emmanouel A. Varvarigos. Statistical analysis and modeling of jobs in a grid environment. *Journal of Grid Computing*, 6(1):77–101, 2008.

[15] Bernardo Costa, Ines Dutra, and Marta Mattoso. Applying reinforcement learning to scheduling strategies in an actual grid environment. *International Journal of High Performance Systems Architecture (Special issue on Parallel and Distributed Systems, Applications and Architectures), Inderscience Publishers*, 2:116–128, 2010.

[16] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001.

[17] Platform LSF Foundations. `http://support.sas.com/rnd/scalability/platform/PSS5.1/lsf7.05_foundations.pdf`.

[18] N. Fujimoto and K. Hagihara. Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pages 391–398, 2003.

[19] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource allocation in the grid with learning agents. *Journal of Grid Computing*, 3, 2005. `http://dx.doi.org/10.1007/s10723-005-9003-7`.

[20] gLite 4. *User Guide*. `http://www.electro.fisica.unlp.edu.ar/eela/docs/gLite-3-UserGuide.pdf`.

[21] B. D. Goodman. *Squeezing SOAP: GZIP enabling Apache Axis, developerWorkss*, March 2003. `http://www-106.ibm.com/developerworks/webservices/library/ws-sqzsoap.html`.

[22] HTCCondor. `http://research.cs.wisc.edu/htcondor/description.html`.

[23] Oscar H. Ibarra and Chul E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. ACM*, 24(2):280–289, April 1977. `http://doi.acm.org/10.1145/322003.322011`.

[24] V. Kant Soni, R. Sharma, M.K. Mishra, and S. Das. Constraint-based job and resource scheduling in grid computing. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 4, pages 334–337, 2010.

[25] Simrat Kaur and Sarbjeet Singh. Article: Comparative analysis of job grouping based scheduling strategies in grid computing. *International Journal of Computer Applications*, 43(15):28–35, April 2012. Published by Foundation of Computer Science, New York, USA.

[26] Ng Wai Keat, Ang Tan Fong, Ling Teck Chaw, and Liew Chee Sun. Scheduling framework for bandwidth-aware job grouping-based scheduling in grid computing. *Malaysian Journal of Computer Science*, 19(2):117, 2006.

[27] A. Kretsis, P. Kokkinos, and E.A. Varvarigos. Developing scheduling policies in glite middleware. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 20–27, 2009.

[28] E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A. Di Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S. Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, and A. Wilson. Programming the grid with glite. In *Computational Methods in Science and Technology*, page 2006, 2006.

[29] Young Choon Lee and A.Y. Zomaya. A grid scheduling algorithm for bag-of-tasks applications using multiple queues with duplication. In *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on*, pages 5–10, 2006.

[30] Jiadao Li and R. Yahyapour. Learning-based negotiation strategies for grid scheduling. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 8 pp.–583, 2006.

[31] Quan Liu and Yeqing Liao. Grouping-based fine-grained job scheduling in grid computing. In *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on*, volume 1, pages 556–559. IEEE, 2009.

[32] M. Maheswaran, S. Ali, H.J. Siegal, D. Hensgen, and R.F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Heterogeneous Computing Workshop, 1999. (HCW '99) Proceedings. Eighth*, pages 30–44, 1999.

[33] Torque Resourse Manager. `http://www.adaptivecomputing.com/products/open-source/torque/`.

[34] M.K. Mishra, R. Sharma, V. Kant Soni, B.R. Parida, and R.K. Das. A memory-aware dynamic job scheduling model in grid computing. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 1, pages V1–545–V1–549, 2010.

[35] Nithiapidary Muthuvelu, Junyang Liu, Nay Lin Soe, Srikumar Venugopal, Anthony Sulistio, and Rajkumar Buyya. A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids. In *Proceedings of the 2005 Australasian Workshop on Grid Computing and e-Research - Volume 44*, ACSW Frontiers '05, pages 41–48, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc. `http://dl.acm.org/citation.cfm?id=1082290.1082297`.

[36] Edgard Quirino Neto. *Performance Analysis of a Grid Infrastructure with an Application in Spatio-Temporal Data Transfers*, June 2012. Thesis.

[37] Zhong-Ren Peng and Ming-Hsiang Tsou. *Internet GIS: Distributed geographic information services for the internet and wireless networks*. Wiley, Wiley:adr, 2003.

[38] PBS Pro. *User Guide*. `http://www.physnet.uni-hamburg.de/physnet/PBSproUG.pdf`.

[39] R. Prodan and T. Fahringer. Overhead analysis of scientific workflows in grid environments. *Parallel and Distributed Systems, IEEE Transactions on*, 19(3):378–393, March 2008.

[40] R. Sharma, V. Kant Soni, and M.K. Mishra. An improved resource scheduling approach using job grouping strategy in grid computing. In *Educational and Network Technology (ICENT), 2010 International Conference on*, pages 94–96, 2010.

[41] Daniel Paranhos Da Silva, Walfredo Cirne, Francisco Vilar Brasileiro, and Campina Grande. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Applications on Computational Grids, in Proc of Euro-Par 2003*, pages 169–180, 2003.

[42] D. M. Sosnoski. *Improve XML transport performance, Part 1, developerWorks*, June 2004. `http://www-106.ibm.com/developerworks/xml/library/x-trans1/`.

[43] D. M. Sosnoski. *Improve XML transport performance, Part 2, developerWorks*, June 2004. `http://www-106.ibm.com/developerworks/xml/library/x-trans2/`.

[44] Inc. Sun Microsystems. *Fast Web Services.* `http://java.sun.com/developer/technicalArticles/WebServices/fastWS`.

[45] Inc. Sun Microsystems. *Fast Infoset.*, 2003. `http://java.sun.com/developer/technicalArticles/xml/fastinfoset`.

[46] Portable Batch System. *Administrator Guide.* `http://www.lxfarm.mephi.ru/docs/v2.3_admin.pdf`.

[47] Srikumar Venugopal, Rajkumar Buyya, and Lyle Winton. A grid service broker for scheduling distributed data-oriented applications on global grids. pages 75–80. ACM Press, 2004.

[48] Zhen Wang and Junwei Cao. Committee-based evaluation and selection of grid resources for qos improvement. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 138–144, 2009.

[49] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, May 1989. `http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf`.

[50] Demetrios Zeinalipour-Yazti, Kyriakos Neocleous, Chryssis Georgiou, and Marios D. Dikaiakos. Failrank: Towards a unified grid failure monitoring and ranking system. In Marco Danelutto, Paraskevi Fragopoulou, and Vladimir Getov, editors, *CoreGRID Workshop - Making Grids Work*, pages 247–259. Springer, 2007.