

Introduction

Aim of this introduction

In this introduction a first general definition of Grid Computing is offered and commented. Then a short overview of the computational grid discussed within this guide is presented. At the moment all the infrastructure are based on EGEE/gLite middleware.

The remaining part of the chapter is dedicated to a short description of the main logical elements and resources forming an EGEE/gLite computational grid.

Contents

- Aim of this introduction
- General Definition
- Computational grid/projects this document is referring to
 - ◆ EGRID
 - ◆ EUINDIAGRID
- Resource types within EGEE middleware/grid.
 - ◆ User Interface
 - ◆ Computing Element
 - ◆ Storage Element
 - ◆ Shared Services
 - ◆ Resource Broker
 - ◆ The Logical Filename Catalog
 - ◆ The Information System
- How to use the GRID

General Definition

A Computer Grid (referred to as Grid) is an infrastructure that enables the integrated and collaborative use of high end computers, networks, databases owned and managed by multiple organizations. It involves the networking of geographically distributed computing resources and makes them transparently available to the end-user.

Grid applications often involve large amounts of data and/or computing resources and may also need secure resource sharing. It is believed that the use of scientific instruments could be incorporated as an additional feature of the existing grid resources, in the nearest future.

The software that glues together all the resources within a grid is the so called "grid middleware". It's generally a quite complex software stack formed by many different layers.

There are many different kinds of middleware deployed by many GRID projects/initiative around the world.

FIXME!

mettere una lista ? -- Stefano

Computational grid/projects this document is referring to

EGRID

The EGRID project is an Italian national Grid infrastructure for financial and economic research implemented by The Abdus Salam ICTP, in collaboration with CNR–INFN Democritos and funded by the Italian Ministry for Education and Research [1.1]2. The EGRID project is deployed based on the EGEE/LCG grid middleware with some interesting and significant add-ons.

FIXME!

mettere una lista ? — Stefano

At the moment the EGRID project is maintaining and operating on three different computational infrastructures:

- ◆ egrid–testbed
- ◆ gridats
- ◆ production grid (within EGEE)

The egrid–testbed is a small infrastructure for just testing/development. Access is granted just to people strictly interacting with EGRID developing team.

The gridats infrastructure is provide within the gridats project. It is therefore available for users belonging to the academic institutions of the Trieste Area.

The EGRID production grid is actually a Virtual Organization (VO) within the EGEE computational infrastructure with its own storage, portal and tools. It is available to all the EGRID official users.

EUINDIAGRID

EUINDIAGRID, started on 1st of october 2006 and sponsored by EU, aims at enabling the interconnection between the most relevant European Grid infrastructure, EGEE, and the Indian Grid infrastructure, GARUDA INDIA.

At the moment EUINDIAGRID is just a Virtual Organization within the EGEE infrastructure.

Resource types within EGEE middleware/grid.

The resources of the EGEE/gLite grids are organized into a number of sites and each site contains a set of machines. Each of these machines is assigned a specific task and depending on their task they are given specific names such as; user interface, computing element, storage element, gatekeeper and worker node, etc. The following is a brief explanation of each of these components.

For a detailed discussion please refer to section 3.2 of the gLite user guide.

User Interface

This machine runs the User Interface (UI) software which allows the end–user to interact with the grid system. This is typically the machine the end–user uses to submit jobs to the grid system and retrieve output

of the completed jobs. The interface is also used to monitor the execution of jobs after submission.

Computing Element

A Computing Element (CE) can be described as one of the following:

- ◆ a gatekeeper machine and a number of worker nodes, or
- ◆ a single queue in the Local Resource Management System (LRMS) or batch system.

A gatekeeper is the front–end of a computing element. It handles the interaction with the rest of the grid environment accepting jobs, dispatching them for execution and returning the output. This provides a uniform interface to the computational resources it manages.

Note

Note:

An actual CE should consist of a gatekeeper machine and a number of worker nodes however; in some sites, you will find only a gatekeeper machine acting also as worker node.

Worker Nodes (WN) sit behind a gatekeeper and are typically managed via a local batch system. The gatekeeper hides the details of WNs from the end–user; however, these are the nodes on which user computations are actually performed. If the Grid enabled applications require any specific software those should be installed on these WNs.

Storage Element

The Storage Element (SE) provides uniform access to large storage spaces. The storage element may control large disk arrays or mass storage systems. This element hides the details of the backend storage systems and provides a uniform interface to the grid user.

Shared Services

The resources within a grid site and the total number of sites change over time as new resources are added to the grid or are temporarily withdrawn for reasons such as maintenance.

There are also several nodes which provide shared services and are not site–specific but shared by various subgroups of the grid users. Resource Broker, File Catalog, Monitoring and Discovery Service and Information Index are some of the common services.

Resource Broker

The Resource Broker (RB) accepts jobs from users (via the User Interface), match the jobs' requirements to the available resources at the various sites within the grid, and dispatch them.

The Logical Filename Catalog

The Logical Filename Catalog (LFC) maintains a database of the locations of master copies of files and the locations of any replicas for a Virtual Organization. They do not hold the actual data only the database describing them (metadata). These machines are used by users and grid services to locate appropriate copies of data files.

The Information System

Globus' Monitoring and Discovery Service (MDS) is a type of infrastructure used to propagate local information to the whole grid. MDS is based on a hierarchy of Lightweight Directory Access Protocol (LDAP) servers. MDS services running on individual grid resources such as CEs² and SEs² collect information about those resources and publish this information into the Information Index (II). In section 3.1.1 MDS will be discussed in detail.

The Berkeley Database Information Index (BDII) is an OpenLDAP² based implementation of an Information Index. BDII is a node which caches MDS information from CEs² and SEs² for use by the Resource Broker and by end-users. BDII could either be top-level BDII and site BDII (Section 3.1.1).

Figure 1.1 depicts a schematic diagram of a computer Grid.

Recently, however, a new type of Information Service called the Relational Grid Monitoring Architecture (R-GMA) has started to be deployed, and many applications are already using it.

Figure 1.1 – A schematic diagram of a computer Grid site

How to use the GRID

In this section we present a typical user interaction with the Grid. Grid computing with the gLite middleware is patterned on the batch processing job execution paradigm; if you are already familiar with batch job processing [1]², then using the Grid entails little more than using a new job submission command.

[1]² e.g., UNIX at/batch, Torque/OpenPBS², LSF.

Let us walk through a typical example.

Alice wants to execute a Monte-Carlo simulation that can last several hours (that is, it is an instance of a *CPU intensive* program), with different boundary conditions; we shall assume, for the example in case, that these "boundary conditions" can be read off a computer file. So Alice wants to use the Grid to execute several instances of the same program, each instance working on a different input dataset.

Alice will then perform the following actions on a gLite/EGEE User Interface, to submit her computational job to the Grid [2]²:

1. Gather the program executable and data
 - ◇ compile the program for the target execution architecture
 - ◇ prepare a shell script to drive job execution
 - ◇ prepare the JDL file to drive job submission
 - ◇ upload data files to the Grid before job submission, if they are too large or too many.
2. Submit the job to the Grid
 - ◇ Create a proxy certificate to authenticate with Grid services.
 - ◇ Submit the job to the Resource Broker server.
 - ◇ Store Job Ids returned by the Resource Broker for later reference.
3. Wait for the submitted jobs to finish.
 - ◇ Monitor job status at regular intervals.

4. Edit and debug failed jobs (if any)
5. Retrieve output data from correctly finished jobs.

We assume that Alice has already adapted her Monte–Carlo program to run non–interactively in her Institution's batch job processing cluster, so we shall highlight only the steps needed to run the same program on the Grid.

[2] Adapting a program to run in a batch job processing system can require some effort, and is arguably the most critical step in the whole process. However, since batch job processing is used since the very early days of computer science, there's already a vast amount of literature on the subject, and we shall not add to it.