Practical No. 12

Grid Services - Study of Grid services using various tools.(any two)

Abstract

Grid computing has emerged as an important new field, distinguished from traditional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation. In this article, we explore this field.
We discuss this area by three sections: first, an introduction: What is grid computing? Why do we need computational grids? Second, a short description of Grid Architecture: How does a grid architecture be built? And, what problems must be solved to make grids commonplace? Third, we use a famous example to illustrate the points mentioned above.

1. Introduction

Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements. The goal of Grid Computing is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources.

Grid Computing is a subset of distributed computing, where a virtual super computer comprises of machines on a network connected by some bus, mostly Ethernet or sometimes the Internet. It can also be seen as a form of Parallel Computing where instead of many CPU cores on a single machine, it contains multiple cores spread across various locations. The concept of grid computing isn’t new, but it is not yet perfected as there are no standard rules and protocols established and accepted by people.

Reason for grid computing

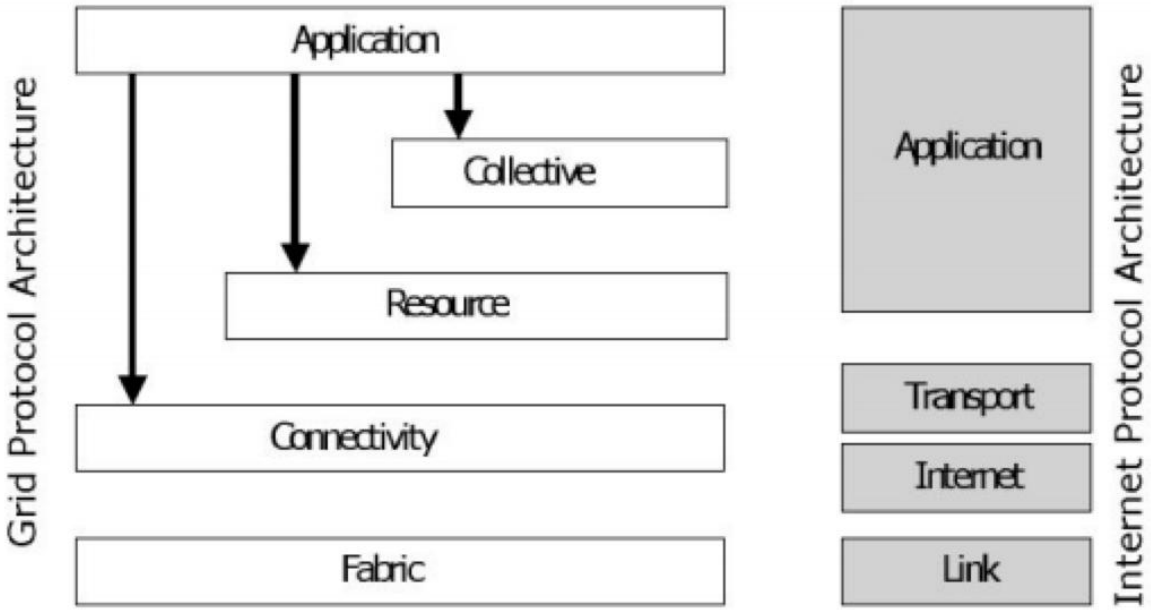
Why do we need computational grids? Computational approaches to problem solving have proven their worth in almost every field of human endeavor. But, there are certainly challenging problems that exceed a computer's ability to solve them, therefore, one important factor is that the average computing environment remains inadequate for such computationally sophisticated purposes. While today's PC is faster than the Cray supercomputer of 10 years ago, it is still far from adequate for predicting the outcome of
complex actions or selecting from among many choices. That, after all, is why super computing environments have continued to evolve.

Advantages of Grid Computing:

1. It is not centralized, as there are no servers required, except the control node which is just used for controlling and not for processing.
2. Multiple heterogenous machines i.e. machines with different Operating Systems can use a single grid computing network.

3. Tasks can be performed parallely accross various physical locations and the users don't have to pay for it(with money).

2. Grid Architecture



Layers of Grid Architecture

- **Fabric: Interfaces to Local Control**
  The Grid Fabric layer provides the resources to which shared access is mediated by Grid protocols.

- **Connectivity: Communicating easily and securely**
  The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources.

- **Resource: Sharing single resources**
  The resource layer defines protocols (and APIs and SDKs) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources.

- **Collective: Coordinating multiple resources**
  The collective layer handles the interactions among a collection of resources. This layer implements functions such as resource discovery, co-allocation, scheduling, brokering, monitoring, and diagnostics.

- **Application:**

The application layer comprises mainly user applications. The applications interact with components in other layers by using well-defined APIs and SDKs.

3. Grid Computing Simulation Tools

GridSim

GridSim toolkit is an open source platform to simulate the grid computing environments and able to model heterogeneous resources. It includes an extensible information system which is able to store and query information about resource properties to design a resource discovery system. It specifies an arbitrary network topology in the simulated grid environment.

Some of the main features of the GridSim toolkit are as following:

a. It allows modelling of heterogeneous types of resources. They also have advanced reservation ability.
b. Applications with different parallel models can be simulated. There is no limit on the number of application jobs that can be submitted to a resource.
c. Multiple user entities can submit tasks for execution simultaneously in the same resource and both static and dynamic schedulers are supported. It allows modelling of several regional grid information services (GIS) components for resource discovery. The characteristics of users in GridSim include job properties (e.g. run time), scheduling constraints, time-zone, budget, etc.

The GridSim has a multi-layered and modular architecture. The first layer is concerned with the scalable Java interface and the runtime machinery, called JVM (Java Virtual Machine), whose implementation is available for single and multiprocessor systems. In the second layer, there is a basic discrete-event infrastructure built using the interfaces provided by the first layer.

Modelling and simulation of core grid entities such as resources, information services, and so on are handled by the third layer and the fourth layer is concerned with the simulation of resource aggregators called grid resource brokers. A broker receives a task issued by a user and applies the scheduling on it. With regard to competitions between users for limited resources, the broker is responsible to handle the related tradeoffs. The final layer is focused on application and resource modelling with different scenarios using the services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms.

Micro Grid

The aim of MicroGrid was to provide a set of simulation tools which are able to establish and evaluate the grid computing middleware, application, and network services. It enables performing scientific experiments as repeatable and controllable. Grid applications can be run on virtual resources and the researchers can study behaviour of complex dynamic conditions. Virtual resources are modelled as a part of overall simulation and they can be of computing or networking types. It explores a set of virtual resources and executes the applications on a set of

physical resources. It has flexibility to handle accurate experiments with heterogeneous physical resources.

The MicroGrid enables using Globus applications by execution virtualization which provides a virtual grid. So the current grid applications can be experimented easily. It virtualizes host identities by mapping each virtual/logical host to one real/physical node using mapping tables. The mapping tables contain virtual IP-address to physical IP-address mappings and are used by libraries. By this way, the programs can be performed as transparent on virtual hosts and they are able to communicate with other virtual hosts' running programs. The users log in on the physical hosts and submit their job to a virtual host. It also supports large-scale network which is critical in grid environments simulation.

The programming framework of MicroGrid is structured and implemented by C and a limited number of statements are provided to simplify the Globus based simulations. Users modify some files of existing Globus applications to perform the simulations. Finally the main goals of MicroGrid are supporting scalable applications by utilizing scalable clustered resources, supporting practical grids software environments to perform the simulated applications accurately, and configuring resources characteristics easily.