5

Design and Implementation of Proposed Algorithms

5.1 Introduction

Chapter 3 and chapter 4 proposed the algorithms for grid resource scheduling and resource discovery. Details of design and implementation of proposed algorithms are discussed in this chapter. Two types of implementations are used to evaluate the proposed algorithms: 1) Grid Simulator based on GridSim toolkit, and 2) Grid Test bed based on the simulation model used Braun et. al.**[**21**].**

5.2 GridSim Toolkit

The GridSim toolkit [122]-[126] supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and

distributed memory machines such as PCs, workstations, SMPs, and clusters managed by time or space-shared schedulers. That means, GridSim can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems such as clusters, Grids, and P2P networks. Application schedulers in Grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. Reference [101] provides a list of the salient features of GridSim. GridSim is a Java-based discrete event simulation toolkit. Java classes represent the entities essential for application, resource modeling, scheduling of jobs to resources, and their execution along with management. Figure 5.1 (taken from [125]) shows the flow diagram of GridSim based simulations. Visual Modeler [127] for GridSim provides a graphical user interface based tool that enables the users to specify inputs required for scheduling like jobs and resource characteristics etc., in an easy manner. A java program is automatically created by the Visual modeler for the simulation that can be executed using GridSim.

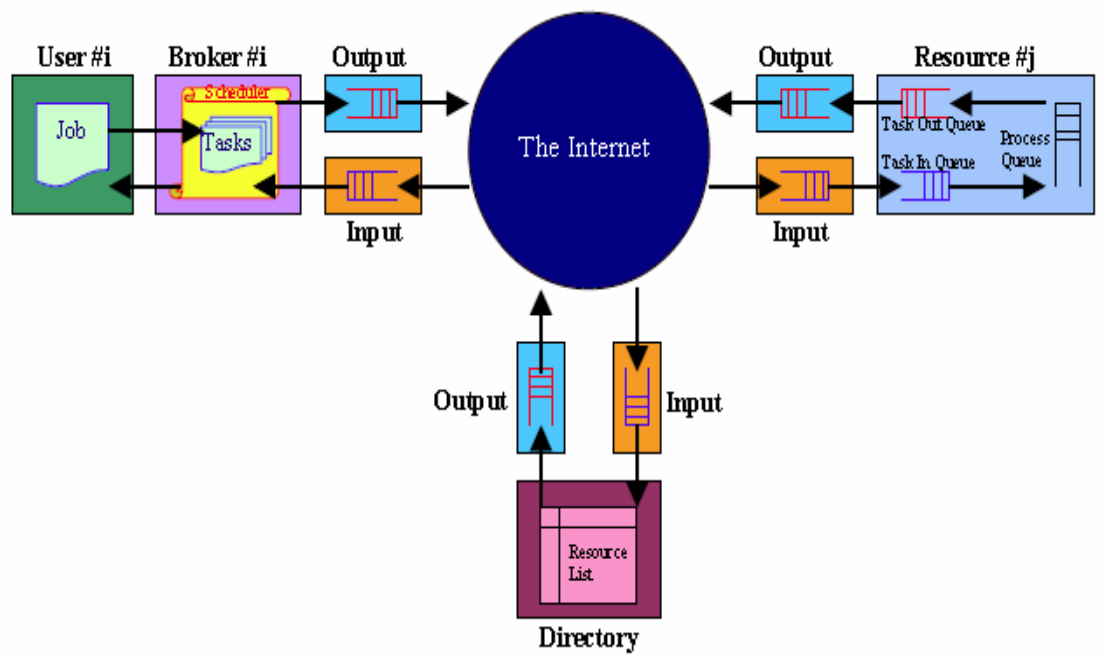

Figure 5.1: Flow diagram in GridSim based simulations [125]

5.2.1 Resource modeling

In GridSim toolkit, CPUs can be created (called as Processing Elements (PEs)) with different MIPS (Million Instructions per Second) or SPEC like ratings. One or more CPUs can be put together to create a machine. Grid resource is composed of such one or more machines. Grid resource can be a single processor, shared memory multiprocessors (SMPs) or a distributed memory cluster of computers and can be managed by time shared or space shared scheduling systems depending on the type of the resource. When a resource entity is created, it registers resource information and contact details to with the grid information service (GIS) entity. The GIS then can be queried for list of resources in a given Grid domain to get resource handles that can be used to query resources directly for their capabilities, costs and other configurations.

5.2.2 Application Modeling

Application model in the GridSim is generally defined by the developers (of schedulers / brokers). Each task in GridSim can be heterogeneous in terms of processing time and input files size, which is defined through a Gridlet object. A Gridlet is a tiny GridApp that contain all the information related to a job/ task and job execution management details such job's processing requirements, disk I/O operations, the size of input files etc. that help in computing the execution time of remote resource and the size of output files. Each grid user can be modeled with different characteristics like type of job created, scheduling policy, activity rate, time zone etc.

5.2.3 Steps for Simulating Application Scheduling

Simulating application scheduling with GridSim involves three high – level steps.

- ✓ First step is the creation of grid resources and gird user. Grid resources that need to be created can have different capability / speed, different time zones and different policies (space - shared or time - shared) as in the real environments. Similarly Grid users can be created with different requirements.
- ✓ Second step is the modeling of applications by creating a number of Gridlets (Gridlets can be considered as grid jobs) and define all the parameters in associated with the jobs. Grouping of Gridlets is also possible depending upon the application model for processing.
- ✓ Finally, last step is the implementation of resource brokers. Resource brokers first inquire the Grid Information System (GIS), then in quire for resource capability including cost and the depending on scheduling heuristics, strategy, or algorithms assign Gridlets to resources for execution. Scheduling policies can be system centric or user centric.

5.3 Implementation details of Grid Simulator based on GridSim Tool kit

Highly dynamic and heterogeneous nature of grid makes it impossible to create a repeatable and controlled environment for experimentation and evaluation of scheduling strategies. Hence a Grid Simulator is developed, which is able to deal with grid scheduling problems like job and resource heterogeneity and runtime changes like arrival of new job, joining / leaving of resources, due to dynamic nature of grid. is This Grid simulator is based on well-known GridSim tool kit This simulator implements a centralized Grid Scheduler, which uses the proposed meta-heuristics based scheduling algorithms (proposed in Chapter 3) for schedule generation.

5.3.1 Components of Grid Simulator

Grid simulator is a modular simulation environment. It consists of independent entities like centralized scheduler, submission system and grid resource, that corresponds to real world entities. Each submission system is associated with it a task generator that is actually used to simulate the job arrivals.

Submission System stores the incoming jobs as they arrive. Also it stores the jobs as they are returned after execution. It communicates with the scheduler to get scheduler information and then selects the resource on to which the job is to be submitted for execution.

Task Generator is associated with the submission system and is used to simulate task / job arrivals. The task arrival times correspond to the selected statistical distribution.

Scheduler is primarily responsible for schedule generation and optimization. Scheduler provides an interface which helps to plug-in the proposed scheduling algorithms for schedule generation. Scheduler has been designed taking the advantages of object oriented paradigm to make it reusable, easy to maintain and easy to integrate with other scheduling algorithms. One part of the scheduler maintains the communication with other components like submission system. Scheduler also implements the functions to estimate the required parameters like makespan, cumulative deadline delay etc. based on the generated schedule and information about the jobs currently under execution.

5.3.2 Class Diagram

Class diagram in Figure 5.2 shows the relation among different classes. *CGridSimulato*r is the main controller class that mimics the grid resource management system. This class contains the information of resources registered with

the grid system. Also, it has the information about the jobs that needs to be executed. Scheduler uses this information about the resources and jobs and estimates the schedule based on the proposed scheduling algorithms.

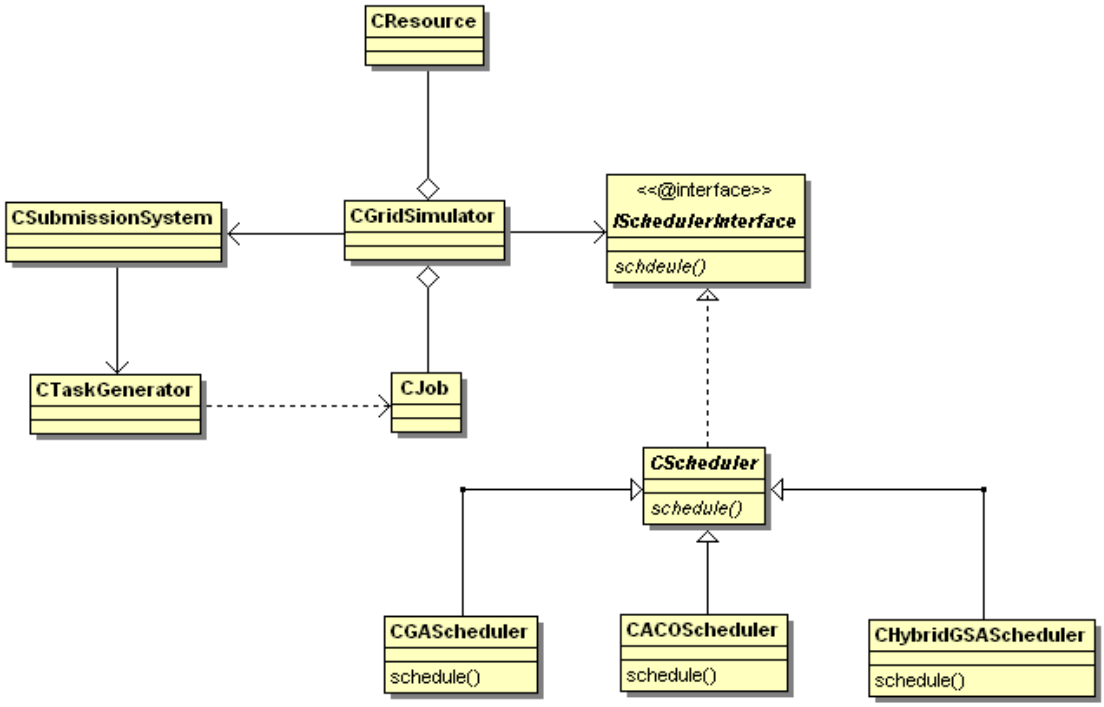

Figure 5.2: Class Diagram of Grid Simulator

CJob and *CResource* classes are used to represent the jobs/ task properties and resource characteristics respectively. Since the main aim of this grid simulator is to evaluate different scheduling algorithms, so this implementation is kept very modular. *CGridSimulator* class provides an interface, *IScheduleInterface*. To add a new scheduling algorithm, this interface needs to be implemented by the any child of the *CSchedule* class. *CSchedule* is an abstract scheduler class to implement the IScheduleInterface. *CGAScheduler*, *CACOSchedudler* and *CHybridGSAScheduler* are the concrete scheduler classes that actually implement the proposed algorithms SGASchedule, ACOSchedule, and HybridGSA respectively. Figure 5.3 shows the screen shot of the development environment for this grid simulator.

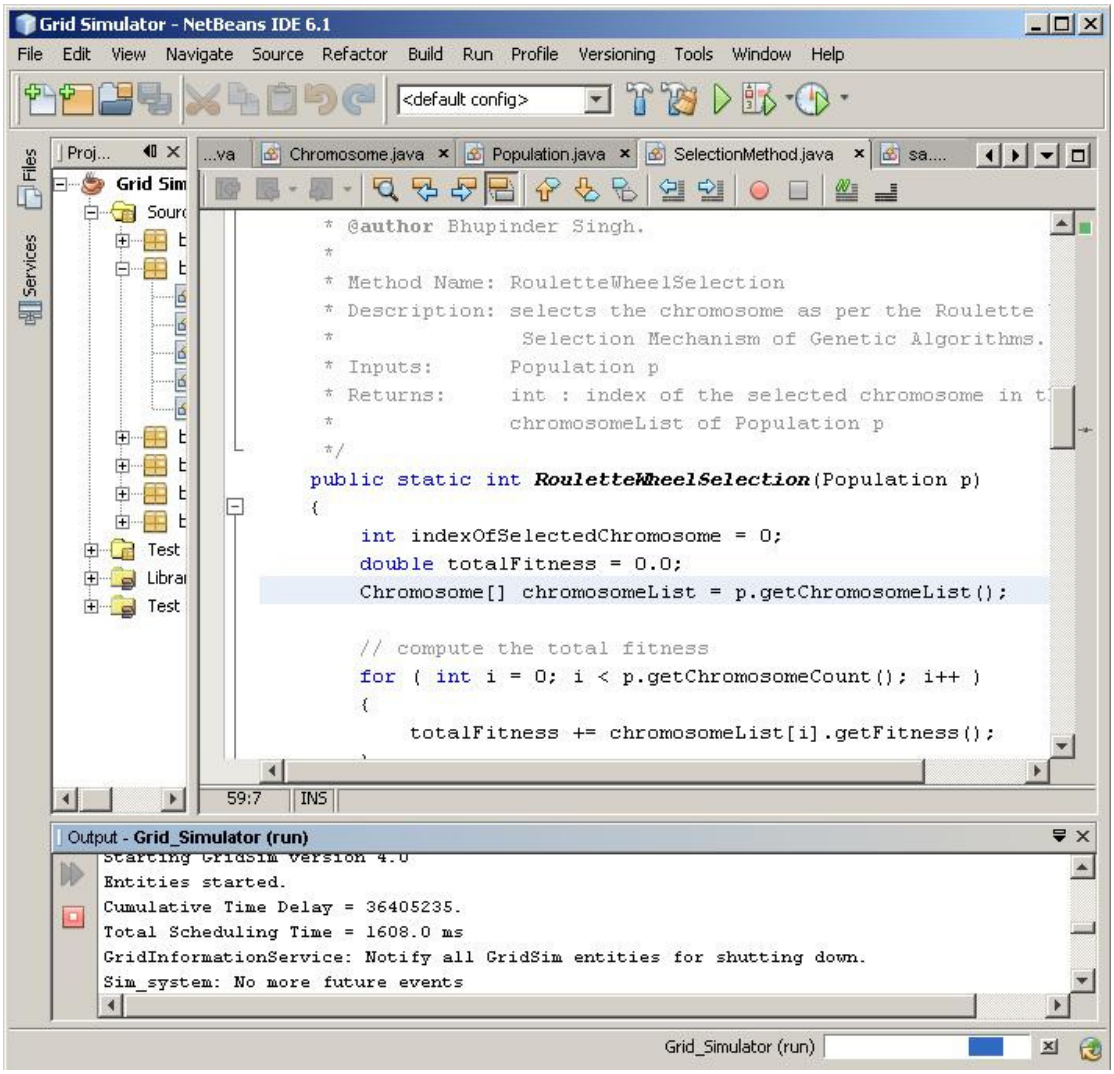

Figure 5.3: Screen shot of development environment for Grid Simulator

5.3.3 Communication Mechanism

The Figure 5.4 shows the communication among submission system, scheduler and grid resource. Submission system submits the job descriptions to the scheduler. The scheduler uses the list of all available grid resources and their characteristics like number of CPUs and their rating. Scheduler has the information about and access to all the available resources registered with the system. On the basis of this information the scheduler generates schedule for each grid resource. Using these schedules and also information of jobs currently in execution the scheduler is

able to approximate various parameters of the schedule such as makespan cumulative time delay for the jobs before their execution and completion.

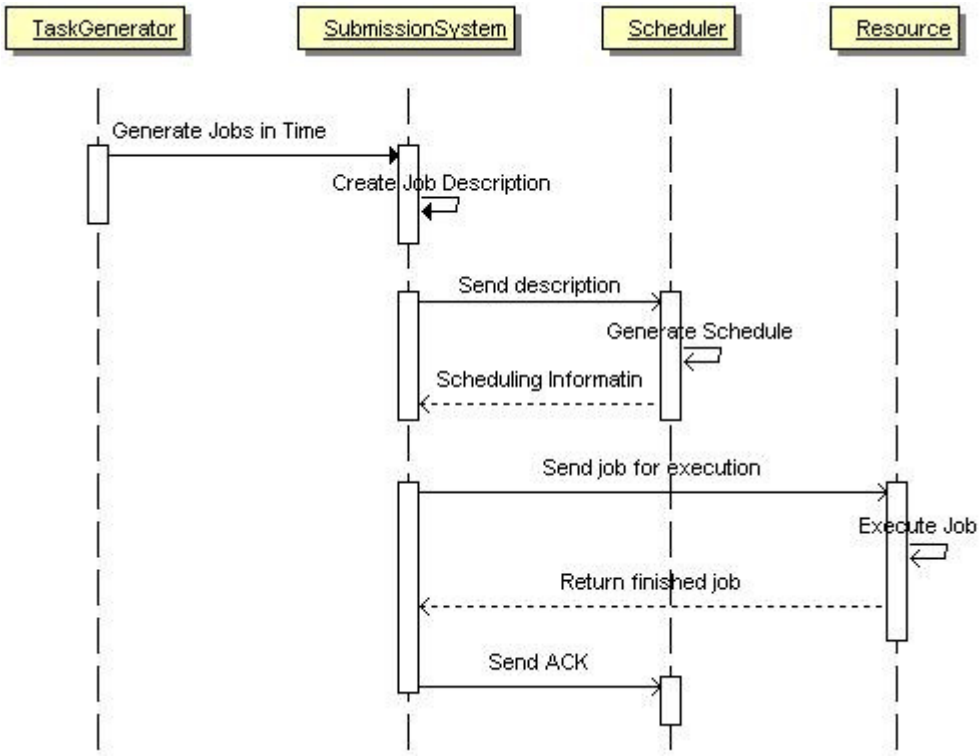

Figure 5.4: Communication among Grid Simulator components

According to the constructed schedule the scheduler responds to the submission system with scheduling information so that submission system is able to know which resource to be selected to execute a particular job. Whole communication among different components is asynchronous and is done via events. As soon as the new jobs are made available by task generator, their description is send by submission system to the scheduler. Submission system does not wait for the scheduler response. As soon as a job is finished an acknowledgement (*ACK*) is send to the scheduler. Scheduler then updates its internal information and current resource load. Pseudo code fro Submission system flow and scheduler flow are given in next sub-section.

5.3.4 Pseudo Code for Submission System Flow

```
1. Create new Gridlets / Jobs
2. Send the job description to scheduler for schedule
generation.
3. Wait for next event
     3.1 ACK (Acknowledgement):
             o Do nothing
     3.2 Gridlet_return:
             o Job execution complete
4. When All jobs finished
             o Send End_of_simulation to Scheduler
```

5.3.5 Pseudo Code for Scheduler Flow

```
1. Get information of registered resources
2. Wait for next event
     2.1 New_Gridlet:
             o Send ACK to Submission system
             o Add to batch of jobs
             o Find Schedule and execute
     2.2 Gridlet_Finished:
             o Lower Resource load
             o Return Job/Gridlet to submission system
     2.3 End_of_Simulation:
             o Get / Calculate Statistics
```

5.3.6 Extensibility

Grid simulator is modular and main functionality is divided into separate parts. So this is easy to extend, modify and maintain. With minor modifications, it is easier to simulate different type of jobs, scheduling algorithm and optimization criteria. To test any new scheduling algorithms, the only thing that needs to be done is to add a new class (e.g. *CNewScheduleAlgo*), which is derived from *CSchedule* class. This new class will override the method *schedule()* inherited from the base class. Similarly, to simulate the job/ task / Gridlet arrival times according to a different statistical distribution, task generator needs to be modified only. The rest of the classes remain intact so the experiments can be repeated with exactly the same setup. All the components interact with each asynchronously via events (refer to pseudo code in the previous sub-sections 5.3.4 and 5.3.5). So any change in the state machine of one

component does is independent and does not impact the state machine of other components. This makes the grid simulator closer to the real world.

5.4 Grid Test Bed

Grid Test bed provides the actual grid like environment with a simple web based user interface for submission of jobs, registering and de-registering of resources and administrative tasks. The Figure 5.5 shows the high level diagram of Grid Test Bed environment.

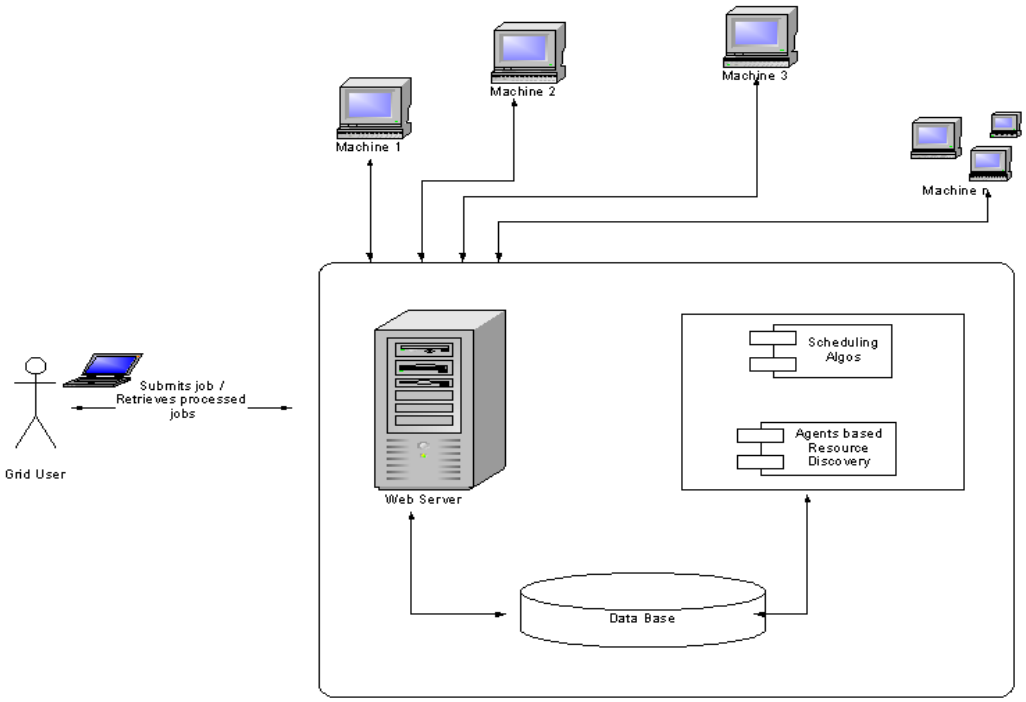

Figure 5.5: High level view of Grid Test Bed used for experiments.

Microsoft Internet Information Server is used as a web server that listens to the user requests provided to the system by web interface. The information provided by the user is stored in the database. User for the system can be grid users or resource owners. Grid users send the job requests for the jobs to be executed. Job characteristics are stored in the database. Grid users also specify the deadline time for the jobs. This is the time before which the job must be executed. Similarly resource

owners use web based interface to register and de-register the resources. Whole information about the resources is also maintained in the database. Scheduling algorithm component provides the implementation of proposed scheduling algorithms. Administrator can change the scheduling algorithms to be used for schedule generation via web interface. The other component is has the implementation of proposed ACO based resource discovery algorithm. All algorithms work on the information present in the database about various jobs and resources and update the information as soon as the job's processing is finished.

5.4.1 Web based User Interface

Web based user interface provides the way for the users to login into the system (Figure 5.6) and submit their jobs for the execution.

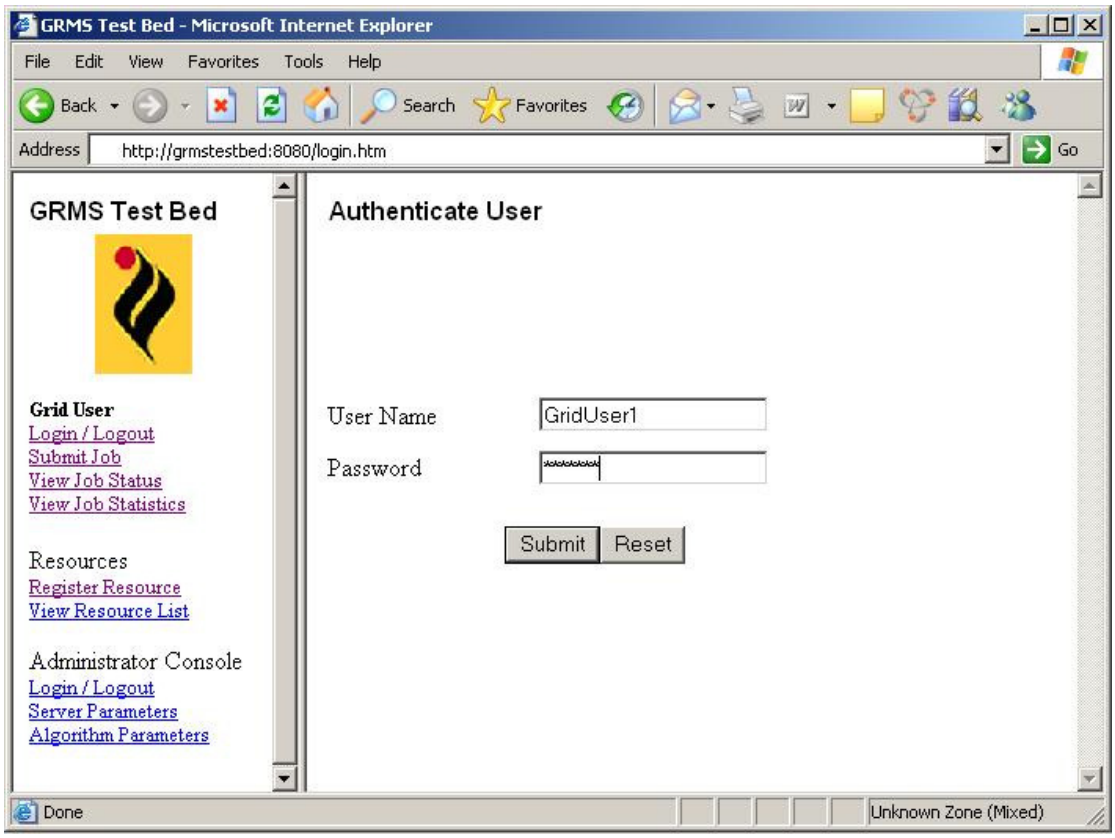

Figure 5.6: Grid User Login Screen

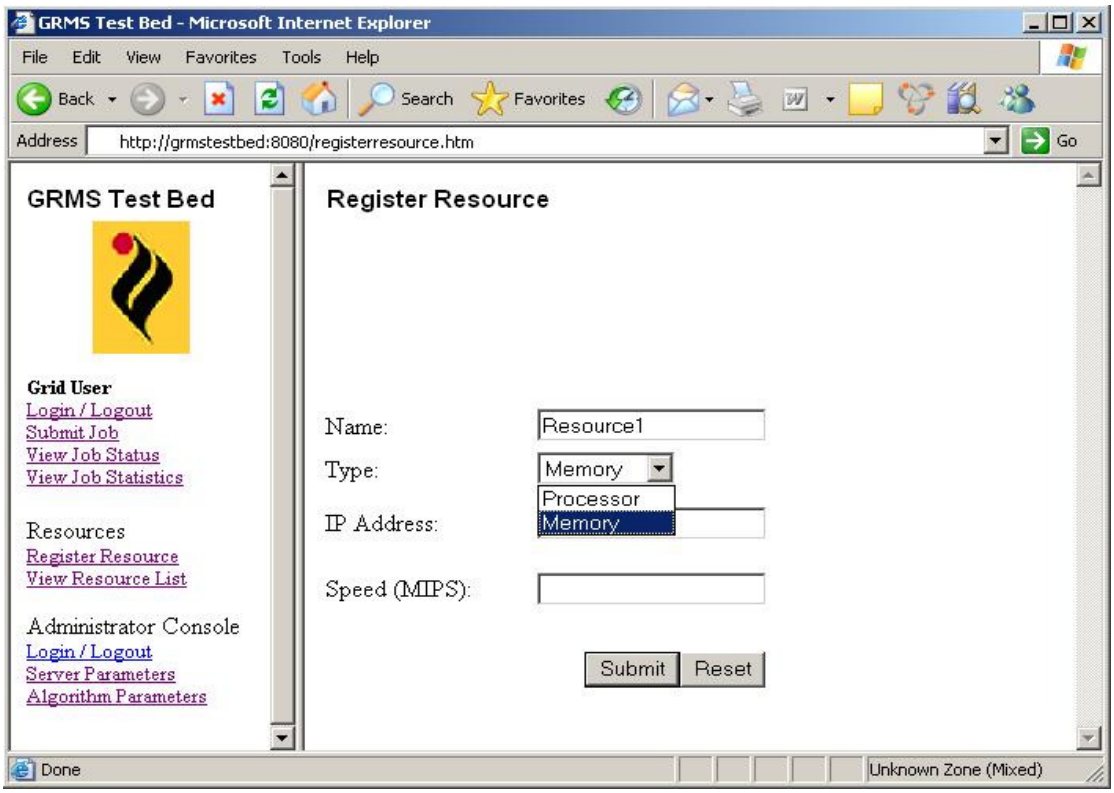

Figure 5.7: Register Resource Screen

.

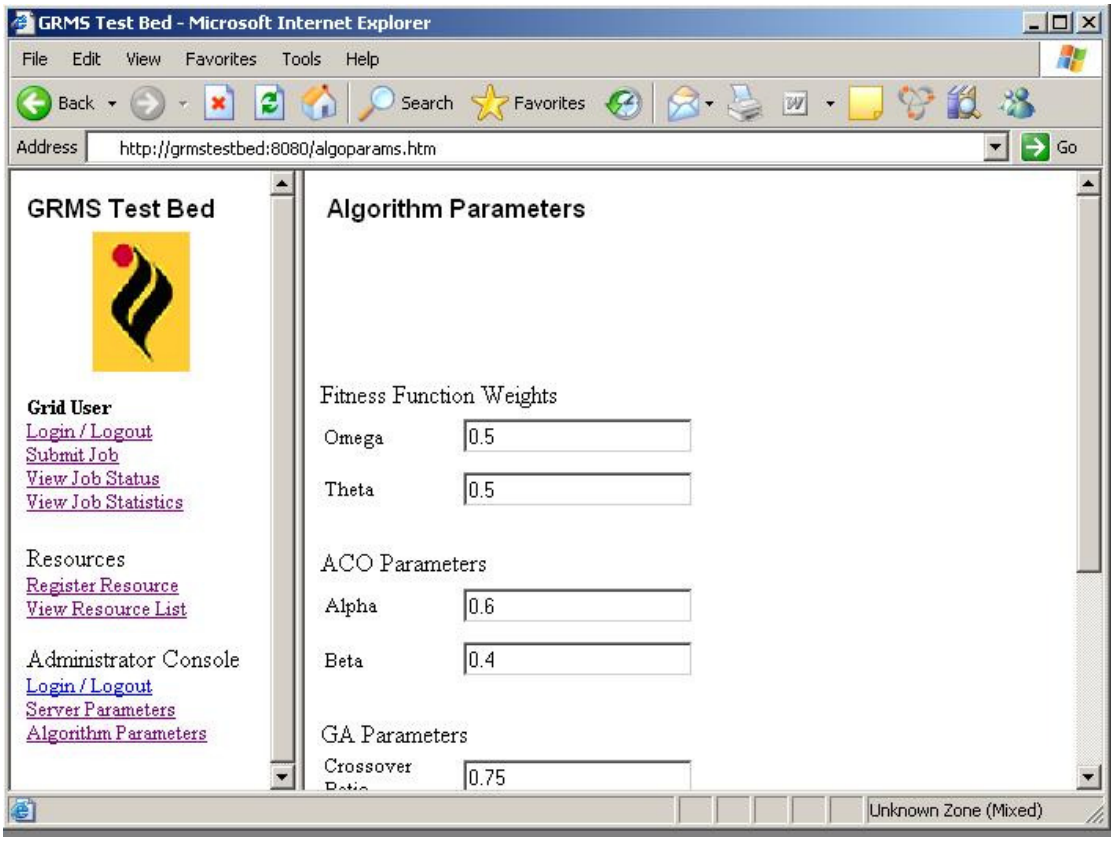

Figure 5.8: Algorithm Parameters configuration screen

The information about the jobs and resources is stored in a database. The Figure 5.7 shows the screen to register resource. Resource discovery algorithms and scheduling algorithms interact with this database to get the information about the jobs and the matching resources so as to generate the schedule. After the schedule generation the resources are contacted via their IP address and the jobs are transferred to the resources for execution. Web interface also provides the way for administrators to configure the system, like log files location, size, server port etc. Administrators can also change the algorithm parameters for proposed algorithms as shown in the Figure 5.8.

5.4.2 Information flow in Grid Test Bed

All components work independently of each other and information sharing is achieved via common database. Database maintains the information about resources, job descriptions and also the different states of jobs and resources. When ever a job execution is completed the information is updated in the database. Grid users can view the status of a job at any time by entering the job IDs. Similarly the statistics for the completed jobs can be viewed by the grid user via web interface by clicking on "View Job Statistics" in the menu bar.

Sequence diagram in the Figure 5.9 shows the high level flow of information and control among various components. A screen shot of the development environment is shown in the Figure 5.10.

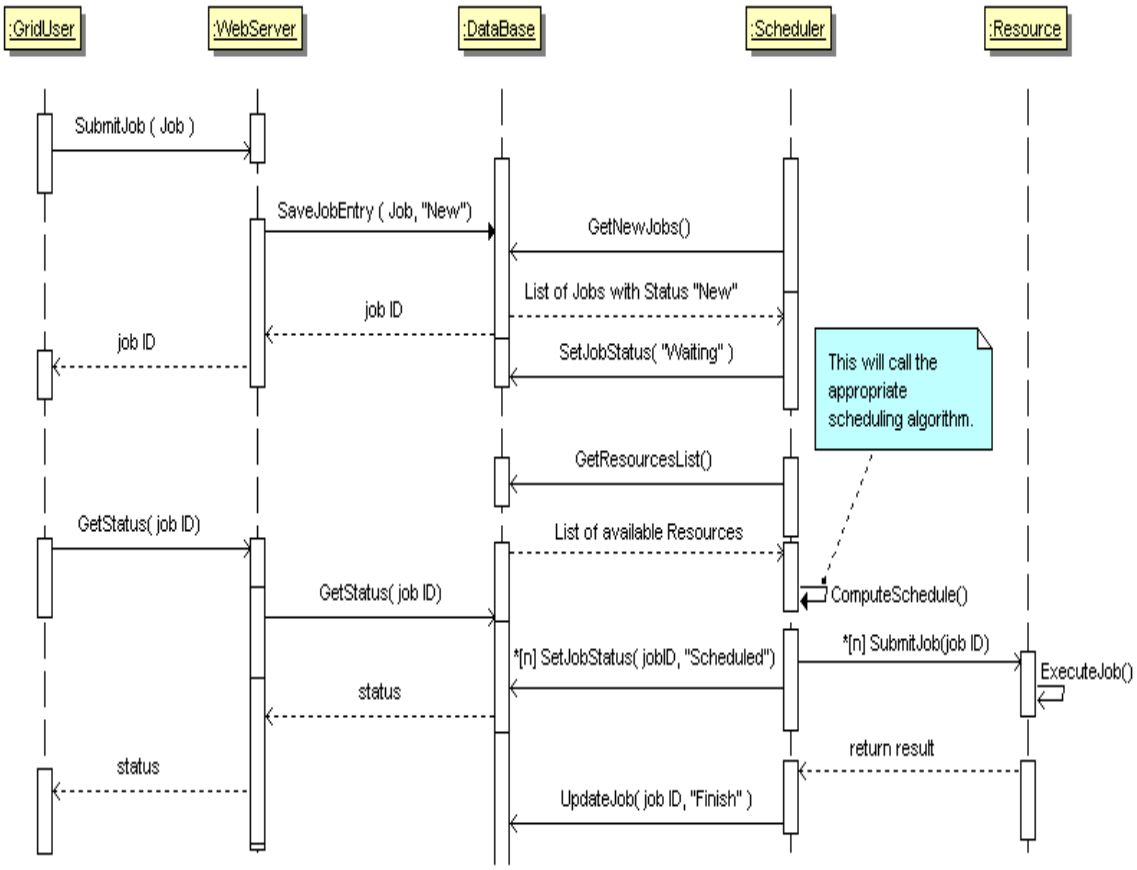

Figure 5.9: Sequence diagram showing information flow in Grid Test Bed

5.4.3 Simulation Model used in Grid Test Bed

This test bed uses the simulation model as used in [21]. Since the results for the most of existing scheduling heuristics are already available, hence it is possible to compare the performance of proposed algorithms with the existing scheduling heuristics. Details of the simulation model are provided in chapter 6 (section 6.2). A completion time matrix (*CT*) is computed based for the job completion times on different resources. The prediction system in the grid test bed uses this completion time matrix to predict the execution time of a job on a resource. *CT* is an $N \times M$ matrix, which contains an entry for every job- resource pair. This entry specifies the completion time for a job on that particular resource.

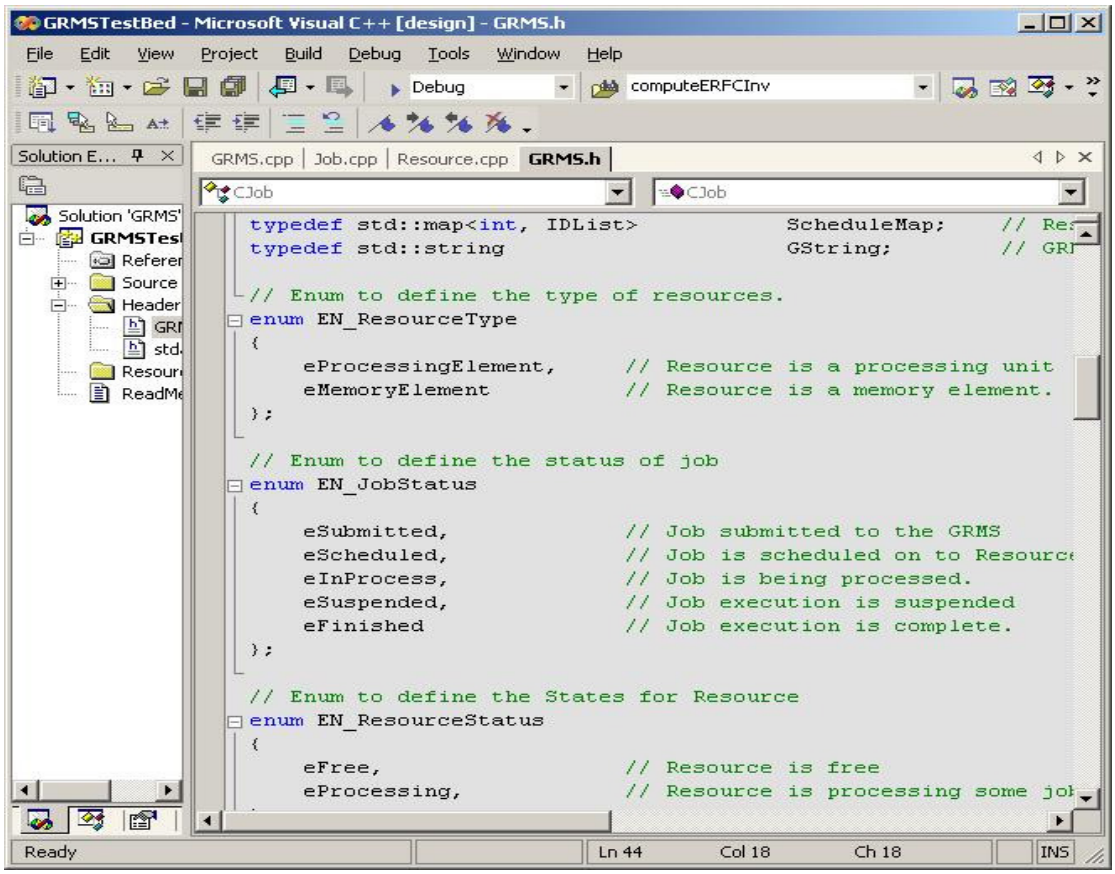

Figure 5.10: Screen shot of development environment in Grid Test Bed

In the actual grid systems state estimation is generally done based on the task profiling and analytical benchmarking. In this experimental test bed, a pre - computed completion time matrix is used, instead of actual task profiling. Completion time matrix can be consistent, partially consistent or inconsistent that corresponds to the homogeneous, semi-heterogeneous, heterogeneous environments respectively. Since grid is highly dynamic and heterogeneous environment, hence only partially consistent and inconsistent matrices are considered for the experiments. Furthermore the heterogeneity can be considered as machine or task heterogeneity. CT matrix can be varied based on low machine and task heterogeneity and high task or machine heterogeneity. Taking into consideration that grid is highly heterogeneous environment; only high machine heterogeneity and high task heterogeneity *CT* matrix are considered in experiments.

5.4.4 Mobile Agent based Framework

To test and validate proposed resource discovery algorithm, a basic framework for mobile agents is implemented in the Grid Test bed. This framework is implemented on the basis of soap services. A soap server is build using the gSoap [128] (a C++ based toolkit for implementing soap services) that listen on a designated port. This server is implemented as a Microsoft Windows service running as a background process on every machine registered with the grid. This soap server publishes the interfaces for Mobile agents transfer from one node to another. Mobile Agent at any node connects to service listening on the designated port on the next node. As soon as the connection is established, the mobile agent calls a method on the remote machine published as a soap method, and travels to the next node. The complete state of mobile agent is serialized and transferred to next node, where it is again de-serialized and mobile agent with the same state is constructed.

5.5 Summary

In this chapter, implementation detail about the grid simulator and grid test bed are provided that implement the proposed algorithms. Grid simulator is GridSim toolkit based environment that implements a centralized scheduler. Grid test bed is grid like environment that is build on the top of simulation model used by the Braun et.al. in their study of scheduling heuristics. The main aim of these implementations is to evaluate and validate the proposed algorithms against a benchmark to demonstrate their usability. Grid Test bed also implements a simple framework for mobile agents. This framework publishes the soap methods for mobile agents travel across various nodes. Mobile agents travel from node to node by serializing and de-serializing their state at source and destination nodes respectively. Next chapter provides the details about the experiments done and their results.