# NGSched - An Efficient Scheduling Algorithm Handling Interactive Jobs in Grid Environment

Madhuri Bhavsar, S.N.Pradhan
*Department of Computer Science & Engineering, Nirma University*
*madhuri.bhavsar@nirmauni.ac.in, snpradhanr@nirmauni.ac.in*

***Abstract***

*A Computational grid is highly useful computing infrastructure that enables effective access of resources to perform high performance computing wherein job management, efficient resource utilization are key grid service issues. To make grid environment greatly useful, an optimized scheduling system is essential as computational grid is often heterogeneous with complex environment. This complexity raises difficulties in gathering the load information of computational resources as well as coping up with user needs specifically when users dealt with jobs. Other existing fact of grid computing is that it is best known for handling batch job rather than interactive job submission. Interactive Job submission is useful in situations when a job requires human intervention to control the job result.*

*This research paper tries to satisfy the requirements of the user by developing user driven scheduler. We have developed a scheduler here after referred as NGSched compatible with Globus and hence provides user on-demand and interactive access to grid resources. It adopts novel approach for scheduling multiple jobs by giving privileges of resource selection to the user. The proposed and implemented algorithm also satisfies the deadlines given by the user for job execution on the selected resources and balances the load on grid nodes. The proposed heuristic also calculates the share of each resource according to capability to handle load. Since this scheduler is user centric, it facilitates the user to monitor the job, cancel the job, schedule on selected resources, suspend and resume the job. Results obtained proved that our innovative algorithm is better and efficient.*

***Keywords:*** *Grid Computing, Resource Selection, Optimized Load balancing, Interactive Job Submission*

## 1. Introduction

Grid Computing is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The coordination between multiple administrative domains leads to complexity & heterogeneity in grid environment. The resources that Grid computing is attempting to integrate are varied. The most common resource is computing cycles provided by the processors of the machines on the grid. The processors can vary in speed, architecture, software platform, and other associated factors, such as memory, storage, and connectivity[1].They also can be of different types like supercomputers, workstations, databases, storages, networks, software, and special instruments, advanced display devices, cache, CPU cycle and so on. Amongst these, resource as CPU cycles is considered by NGSched for provisioning, job execution and balancing the load. The Grid also includes coordinated problem solving,which is beyond simple client–server paradigm, leading to more challenges for handling jobs[2]. Users who are the resource consumers adopt the strategy of solving their problems at low cost within a required

timeframe [17]. The resource providers adopt the strategy of obtaining best possible return on their investment while trying to maximize their resource utilization and hence efficient scheduling mechanisms plays key role.

An organization can lease selected resources to solve large computation problems. Resources owned by various administrative organizations can be shared under locally defined policies satisfying the criteria defined and adopted by user as well as resource owners. Harnessing unused CPU cycles can give huge benefits to corporations and provide cost benefits too[3]. To achieve the promising potentials of computational Grids, an effective and efficient scheduling strategy is defined and implemented as stated further in this paper. If a set of standalone jobs are to be executed with no interdependencies, then a specialized scheduler may not be required. However, if it is desired to reserve a specific resource or to ensure that different jobs within the application run concurrently (for instance, if they require inter-process communication), then a job scheduler should be used to coordinate the execution of the jobs[4]. Prioritizing user and his expectations in the form of deadline achievement for completion of jobs and opportunity for resource selection for deployment of jobs are handled by our user centric scheduler. These tasks are done by algorithm by balancing the computational cycles load in the grid environment.

## 2. Related Works

A set of algorithms has been designed by researchers to identify best resources and schedule Jobs to heterogeneous computing systems. Scheduling algorithm employed in [6] describes the mean load based on task predictive execution time as heuristic information to obtain an initial scheduling strategy and then reduce the load of the machine with higher-load at the fastest speed. Min-min algorithm [14] starts with a set of all unmapped Jobs. The completion time for each job on each machine is calculated. The machine that has the minimum completion time for each job is selected. Then the job with the overall minimum completion time is selected and mapped to the machine. Again, this process is repeated with the remaining unmapped Jobs.Min-min[13] begins with a set of all unmapped Jobs. The completion time for each job on each machine is calculated. The machine that has the minimum completion time for each job is selected. From the set, the algorithm maps the job with the overall maximum completion time to the machine. Again the above process is repeated with the remaining unmapped Jobs. Similar to Min-min, Max-min also considers all unmapped Jobs at a time. [7] contributes with the mean of all machines completion time is taken. The machine whose completion time is greater than the mean value is selected. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value. Scheduling is a process that maps and manages execution of inter-dependent tasks on distributed resources. Scheduling algorithm implemented by [9] provides distinct functionalities, such as dynamic workflow planning and just-in-time scheduling in a grid environment. Serve On Time algorithm proposed in [11] is based on the idea of combining all the dynamic multi tasks so that all the tasks will obtain the rights to be served as soon as possible The highly effective dispatch strategy or algorithm may fully use the processing ability of the grid system, and then enhances application procedure performance. [22] Proposes workflow based scheduling algorithm. Workflow is concerned with the automation of procedures, whereby files and other data are passed between participants according to a defined set of rules in order to achieve an overall goal. Developers of Nimrod-G broker[15] claimed that this is the only system that supports resource allocation and application scheduling algorithms driven by users' quality of service requirements. Paper[10] describes Genetic algorithm which uses the GA's searching technique to find a

near optimal schedule in the grid computing environment. In order to realize the necessities required during the usage of grid, number of research priorities have been defined with the focus on Architecture, design and development of the next generation grid. Scheduling problem is an important issue in a grid computing environment, because of the heterogeneity of computing resources[5]. Need of developing algorithms for scheduling of data intensive workflows has been identified by [12].Also vital issues like load balancing, resource management, handling use scenarios, and maintaining security issues are also taken up by researchers. Our attempts in implementing these issues are projected to develop a Globus compatible scheduler as Globus is widely used grid middleware.

## 3. Preface of proposed research algorithm

This section, presents the objective and preface of job scheduling in heterogeneous computing environment. Each machine executes Jobs in round-robin manner. For initial level of mapping, the size of the Jobs and the number of machines in the heterogeneous computing environment is known a priori. Otherwise the accurate estimate of the expected execution time for each Job on each machine is known a priori to execution. Efforts are taken to estimate the required execution time for a job using CASE tool.

This section, presents the objective and preface of job scheduling in heterogeneous grid computing environment. Job is a group of tasks (modules) where each task can be allocated a separate resource as per the selection which is done by the user.

Before scheduling the jobs, following are the prior factors during scheduling the jobs:

- It is expected that the submitted jobs are grid enabled. Each job can have number of modules to be deployed on the resources.
- Numbers of machines are readily available to participate in the allocation of jobs. It is possible that they are having their existing workload.
- The workload of each Job is in millions of instructions.
- The resources are selected by each job.

The deadline of each Job is specified by the user are observed by an algorithm. If in case, the deadline doesn't met, then intimated to the user. These decisions are taken prior to deployment of jobs. Architecture & Design of proposed system is shown in figure 1.

## 4. Proposed Scheduling Algorithm

Job scheduling system is the most important part of grid resource management system. The grid scheduler must make best effort decisions and then submit the task to the hosts selected, generally as a user[8].The scheduler receives the job request, and chooses appropriate resource to run that job. User is also privileged to select the resource if user desires. In this paper, the formulation of job scheduling is based on managing expected time which is specified as a deadline by the user and to achieve Load balancing.

Once all the new Jobs have been submitted, the Scheduling algorithm deployed on NirmaGrid [Grid test bed configured on campus] works in the following phases:

**Algorithm (NirmaGrid):**

*1) Get_from_user (Job_info)*
*2) Get_from_resource (MIPS)*
*3) // Find total CPU cycles / jiffy for all resources*
*for i = 0 to total_resources*
*Total_share = Total_share + Resource MIPS*

```
4)  // Find percentage share of each resource
    for i = 0 to total_resources
    Percentage_share = Resource_share / Total_share
5)  // Find current status of all running Modules
    for i = 0 to total_jobs
    if Job scheduled
    for j = 0 to total_modules
    if Module running
    Get_from_scheduled_resource (Execution_time_left)
6)  // Find total CPU cycles required by all Modules
    for i = 0 to total_jobs
    for j = 0 to total_modules
    Total_CPU_cycle = Total_CPU_cycle +
    estimated_execution_time
7)  // Find CPU cycle share of each resource
    for i = 0 to total_resources
    CPU_cycle_share = Total_CPU_cycle * Percentage_share
8)  // Find Modules that can be scheduled over some resource
    for i = 0 to total_jobs
    for j = 0 to total_resources_selected
    Insert the information of all Modules of the Job in resource structure
9)   Deduct the CPU cycle share of Modules running over resource
    for i = 0 to total_jobs
    for j = 0 to total_modules
    if module running
    deduct Current_execution_completed from CPU_cycle_share of the resource over
which Module is scheduled
10) Sort resources as per CPU_cycle_share
11) Sort Modules of each resource as per Execution_time_left
```

```
Schedule_1://----find resource with only 1 module mapping possible, allocate module to
resource, and update matrix of modules to be scheduled
12) for i = 0 to total_resources
    if total modules that can be scheduled = 1
    for j = 0 to total_modules_running_over_resource
    calculate new deadline
    if deadline not satisfied
    do not schedule the Module
    goto Schedule_1
    calculate deadline for the new Module
    if deadline satisfied
    schedule the Module over resource
```

```
Schedule_2:       //Max-Min
13) for i = 0 to total_resources
    if CPU_cycle_share > 0
    for j = 0 to total_modules
    if CPU_cycle_share > Execution_time_left
    for j = 0 to total_modules_running_over_resource
    Calculate new deadline
    if deadline not satisfied
```

*do not schedule the Module*
*goto Schedule_2*
*Calculate deadline for the new Module*
*if deadline satisfied*
*Schedule the Module over resource*

***Schedule_3: Max-min without deadline***
*14) for i = 0 to total_resources*
*For j = 0 to total_modules*
*For j = 0 to total_modules_running_over_resource*
*Calculate new deadline*
*If deadline not satisfied*
*Do not schedule the Module*
*goto Schedule_3*
*calculate deadline for the new Module*
*if deadline satisfied*
*schedule the Module over resource*

**Figure 1 Proposed Algorithm of NGSched**

## 5. Implementation

Forming Test bed for verification and execution of scheduler using Globus on the campus was challenging task. For configuration of testbed guidelines are referred from [16-21]. The implementation is alienated among various modules. The modules are:

- Providing Graphical User Interface on grid console to obtain interactivity
- Fetching data from MDS
- Fetching data of processes running over remote node
- Maintaining Input & Output between remote node and admin node
- Deployment of Job-Modules as per scheduling decision on remote node
- User centric Job management functionalities

### 5.1. Fetching Data from MDS:

Globus provides Monitoring and Discovery Services to view the current status of all the nodes connected in the Grid. For the design of the scheduling algorithm various CPU parameters like MIPS (Million Instructions Per Second) is required. The MDS provides the list of resources and its current utilization leaving the identified parameter. So these values are acquired by scripting in the grid. From MDS other parameters are extracted and thus the required node details are composed.

### 5.2. Maintaining Input/Output Between Admin and Remote Node:

Globus is used to submit a Job in batch mode. Globus also provides the functionality to submit a Job in interactive mode. But while submitting a Job in interactive mode, the standard input/output of the Job are not redirected to the allocated resource console from where the user has submitted a Job.

PIPE can be used for redirecting standard input/output. But such redirection over network is not possible using PIPE. Hence, a socket connection is established between the admin node and the remote node. And a PIPE is used on the remote node to for input/output redirection capturing input from socket and sending the output to the socket.

### 5.3. Deployment of All Job-Modules over Remote Node:

The GRAM APIs of Globus provides the globusrun -ws which is used to execute a Job. After submitting a Job, the GRAM internally calls Gatekeeper Daemon which creates a Job Manager and also creates a child process which executes the given Job. Using Globus API, the Gatekeeper Daemon creates the Job manager on the specified hostname. And also creates a child process which executes the Job on the specified hostname. The interactivity and Job executable is combined in a new executable and then job is signaled to be executed on the remote node.

### 5.4. User centric Job management functionalities:

User is very important and core factor in grid technology. This project gives privileges to the grid user needed for managing and ruling the jobs submitted on grid. User can -

- Terminate execution of a Job-Module
- Suspend execution of a Job-Module
- Resume execution of a Job-Module
- Monitor the job

User can request to terminate, suspend and resume the execution of a Module using the GUI provided to the user.

## 6. Performance Analysis

Functional activities are handled by multiple agents in the proposed system. Post implementation analysis for verification of functionalities of scheduling algorithm is described below.

### 6.1. Comparing Wastage of CPU Cycles:

The Graph in fig 2 shows a comparison between the FCFS algorithm and proposed algorithm, NGSched, based on the number of CPU cycles wasted in % as the number of Jobs increases. The waste of CPU cycles is calculated by considering the ideal Load of the Resources and the Actual Load of the Resources. As seen in the figure 2, NGSched achieves optimal Load balancing thus not wasting much CPU cycles. Here for analysis, the standard Job is considered with having 2 Modules and having 33% and 66% of the total Load in each Module. The deadline of each Job is considered to be high to submit all the Jobs to analyze the performance of each algorithm.

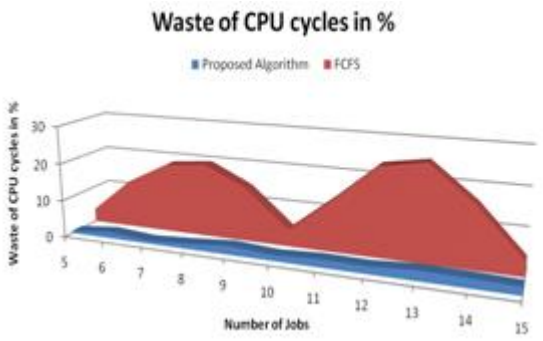


**Fig 2 Minimum Wastage of CPU Cycles by Proposed Algorithm**

### 6.2. Total Jobs Scheduled:

Figure 3 shows a comparison between FCFS algorithm and proposed algorithm, NGSched, based upon the total number of Jobs scheduled from the pool taking in to consideration deadline of the Jobs and achieving Load balancing as per Ideal Load of each Resource. Graph shows as the number of Jobs increases, the deadline considered for each Job is also increased. Here some improvement in the number of Jobs submitted by NGSched is seen as per the strategy of Load balancing over each Resource.

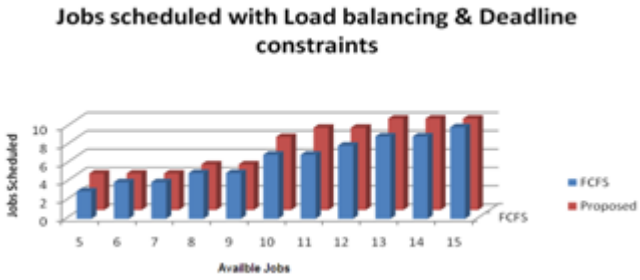


**Fig 3 NGSched Schedules More Jobs**

### 6.3. Jobs Submitted with Given Deadline:

In the figure 4, analysis of the relation between the numbers of jobs submitted with the given deadline is shown. The total number of Jobs is considered to be the same for the analysis. As the deadline of the Jobs increases, the total number of Jobs submitted also increases.

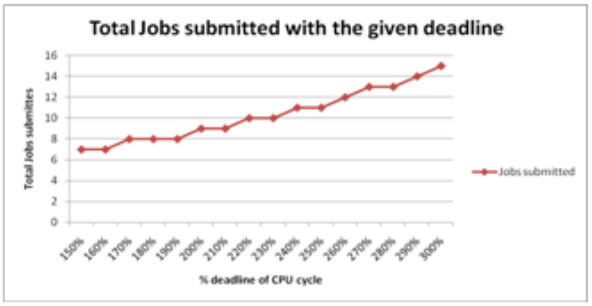


**Fig 4 Job Submission with Maintaining Deadline**

### 6.4. Utilization of Grid:

In Figure 5, analysis of Grid utilization is shown. As the number of Jobs increases, the % utilization of Grid also increases. Each Job has approximately 1500 jiffy of execution time. Thus with the proposed scheduling policy, 100% utilization of Grid can be achieved.

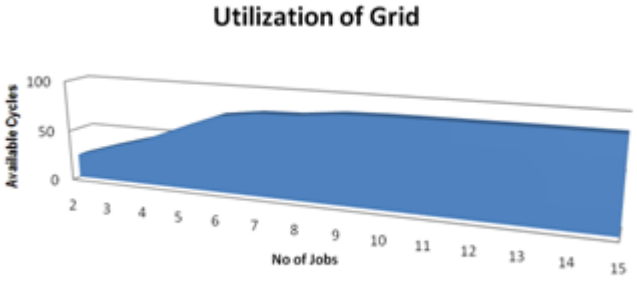


**Fig 5 Utilization of Grid**

### 6.5. Number of Jobs Scheduled:

In Figure 6, the analysis of percentage of jobs scheduled with the deadline constraints is shown. The deadline is considered to be 250% of the execution time of one Job. As the no. of Jobs increases, the total no. of Jobs scheduled decreases.


**Fig 6 No of Jobs Scheduled**

### 6.6. Time Taken for Scheduling Decision:

In Figure 7, the time taken for taking the decision of scheduling is shown. As number of jobs increases, time taken by NGSched is also consumed more.

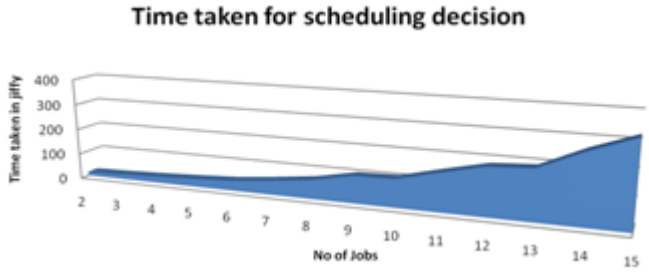


**Fig 7 Decision Measures**

## 7. Conclusion and Future work:

NGSched is intended to provide the interactive benefits and facilitates for grid users and owners. It is compatible with Globus, where currently Globus uses PBS,CONDOR & SGE. NGSched specifically provides an interactive mode of job handling which service maximum jobs (concurrently) in a given time with fulfilling deadline requirement of user. Predicting to the demand of more and more resources, NGSched provides best utilization of (active and idle resources in shared environment) of resources. By providing an interactivity for handling the jobs in complex grid environment, key services provided by NGSched are,to maintain the deadline & balancing the load for optimum utilization of resources.

Further provisions like advanced reservation of resources and response time can be taken into account for more services of NGSched.

## References

[1] Ferreira L., Berstis V., Armstrong J., Kendzierski M., Neukoetter A., Takagi M., Bing-Wo R., Amir A., Murakawa R., Hernandez O., Magowan J. and Bieberstein N., Introduction to Grid with Globus, IBM redbooks.

[2] Li M. and Baker M., The grid: core technologies, Wiley Publications, 2005

[3] Silva V., Grid Computing for Developers, Charles River Media publications, 2006

[4] Armstrong J., Neukoetter A., Takagi M., Bing-Wo R., Amir A., Murakawa R., Ferreira L., Berstis V., Kendzierski M., Hernandez O., Magowan J. and Bieberstein N., Enabling Applications for Grid Computing with Globus, IBM redbooks.

[5] Kun-Ming Yu, Cheng-Kwan Chen, "An Evoluation based Dynamic Scheduling Algorithm in Grid Computing Environment,", 2008 Eighth International Conference on Intelligent Systems Design and Applications

[6] Lina Ni, Jinquan Zhang, Chungang Yan, Changjun Jiang, A Heuristic Algorithm for Task Scheduling Based on Mean Load, First International Conference on Semantics, Knowledge and Grid (SKG'05),

[7] Nov 27-29, Beijing, China

[8] Kamalam.G.K, Muralibhaskaran.V, A New Heuristic Approach: Min-Mean Algorithm for Scheduling Meta-Tasks on Heterogeneous Computing Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010

[9] Stefka Fidanova, Simulated Annealing for Grid Scheduling Problem, IEEE 2006 International Symposium on Modern Computing (JVA'06)

[10] Moohun Lee, Janguk In, Sunghoon Cho, Changbok Jang, Euiin Choi, A Scheduling Middleware for Scheduling on a Grid Environment, IEEE 2006 International Conference on Hybrid Information Technology (ICHIT'06)

[11] Ajith Abraham, Rajkumar Buyya and Baikunth Nath, Nature's Heuristic for Scheduling Jobs on Computational Grids, In Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, (ADCOM2000),

[12] Liying Zhu, Zhengyu Sun, Wei Guo, Yaohui Jin, Weiqiang Sun, Weisheng Hu, Dynamic Multi DAG Scheduling Algorithm for Optical Grid Environment, Proceeding. of SPIE Vol. 6784 67841F-2

[13] Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi, Ji Bian, A Data-Intensive Workflow Scheduling Algorithm for Grid Environment, 2009 Fourth ChinaGrid Annual Conference

[14] Xiaopeng Yu, Xiaogao Yu, A New Grid Computation-based Min-Min Algorithm, 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery

[15] Cho-Chin Lin and Chun-Wei Shih, An Efficient Scheduling Algorithm for Grid Computing with Periodical Resource Reallocation, IEEE ,CIT2008

[16] Rajkumar Buyya , Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis, pp 37-38

[17] http://www.globus.org

[18] https://www.globusconsortium.org/tutorial

[19] http://www.globus.org/toolkit/docs/development/4.1.0/key/index.html/GT4 Primer 0.6.pdf
[20] http://www.globus.org/toolkit/docs/development/4.0.1/
[21] http://www.gridway.org/
[22] http://www.linuxhomenetworking.com/
[23] Jia Yu, Rajkumar Buyya and Kotagiri Ramamohanarao,Ed. Of chapter 'Workflow scheduling Algorithms for Grid Computing,, Springer Publication on Metaheuristics for scheduling in distributed computing environment.

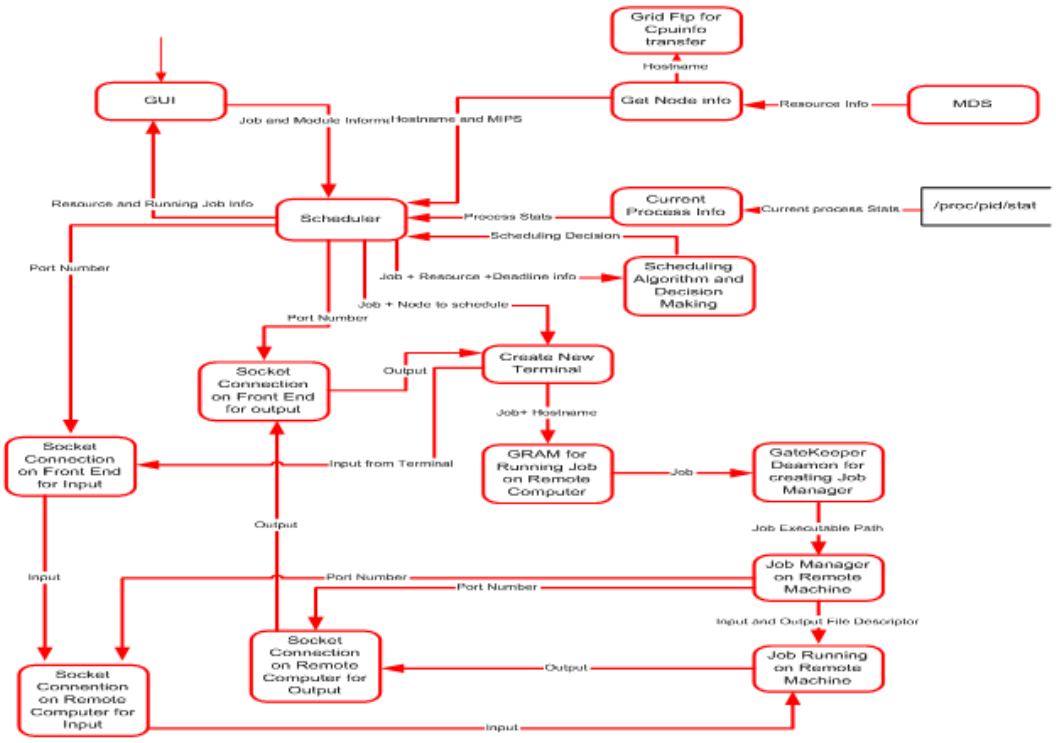


**Figure 8 System Design and Architecture**