CHAPTER 6

JOB SCHEDULING WITH ECONOMIC PARAMETERS
AND FAILURE HANDLING STRATEGIES

In a distributed environment, on one hand, there is a suite of computational resources interconnected by networks; on the other hand, there is a group of users who submit applications for execution on the suite of resources. The scheduling system of such a distributed computing environment is responsible for managing the suite of resources and dealing with the set of applications. In case, if a set of applications wait for execution, the scheduling system should be able to allocate appropriate resources to the applications, attempting to achieve some performance goals. However, a computational Grid has more diverse resources as well as more diverse applications.

In Grid computing, as resources are distributed in multiple domains in the Internet, not only the computational and storage nodes but also the underlying networks connecting them are heterogeneous. The heterogeneity results in different capabilities for job processing and data accessing. In traditional parallel and distributed systems, the computational resources are usually managed by a single control point. The scheduler not only has full information about all running/pending tasks and resource utilization, but also manages the task queue and resource pool. Thus it can easily predict the behavior of resources, and is able to assign tasks to resources according to certain performance requirements.
A distributed resource scheduling algorithm has the capability of handling multiple resource requirements for jobs that arrive in a Grid computing environment. In the resource scheduling algorithm, both the site capabilities and the resource requirements of jobs are considered. The main objective of the algorithm is to obtain a minimal execution schedule through efficient management of available Grid resources. As the existing system considers only two basic dimensions, it needs to be extended to include economic parameters and methods to predict possible failures before execution.

6.1 SCHEDULING ALGORITHM FOR MULTIPLE REQUIREMENTS

Existing multi-dimensional scheduling algorithm by Khoo et al. (2007) has the capability of handling two resource requirements for a job that arrive in a grid computing environment. A job that is submitted in the environment is with computation and data requirements. These two major requirements are needed for effective execution of the job. Computational resources are aggregated to form computation index. Computation and data index determines a good resource site that best executes a job based on I/O. These two indices (computation and data) are calculated for every site in the environment depending on the job’s requirements. A low index value represents the best resource. The existing system considers only computation and data as a dimension for scheduling. It does not consider economic parameters. So there is no provision for user to specify the time and cost within which user’s application must finish.

In the proposed system, in addition to existing two dimensions [computation and data], additional two dimensions are included viz. deadline and budget. User requirements are now based on computation, data, deadline and budget for completion of a job. The proposed scheduling algorithm which
considers the four dimensions is henceforth referred as 4D Scheduling Algorithm and it is constrained to meet all the necessary requirements. A 2D virtual map is plotted for the dimensions considered for scheduling. Only the resources that match deadline and budget are included in the virtual map. The Euclidean distance from the origin denotes the best possible resource that matches the resource requirements of a job for an instance in time. Therefore the resource site which is lying nearer to the origin is the best resource for the job to be submitted. This is because only the closest resource has low deviation from the requirements. This algorithm is analyzed using two parameters namely, average waiting time and queue completion time.

Fault tolerance is an important property for large scale computational grid systems, where geographically distributed nodes cooperate to execute a task. In order to achieve high level of reliability and availability, the grid infrastructure should be fault tolerant. During execution, a job may fail due to resource unavailability. In a reactive failure handling, once a resource fails, the job is rescheduled to some other available node for execution. This job migration needs checkpoint recovery and incurs rescheduling overhead. In a pro-active failure handling, the resource availability is predicted before scheduling a job and the job is dispatched to the resource with the hope that it does not fail. The pro-active failure handling mechanism similar to Khoo and Veeravalli (2010) is incorporated into the proposed 4D resource scheduling algorithm. The 4D scheduling algorithm along with the failure handling mechanism is henceforth referred as 5D scheduling algorithm. Availability index is calculated for each resource in the environment. This availability index gives the probability of resource in the UP (available) state. A 2D virtual map is extended to a 3D virtual map with availability index as an additional dimension. The resource which is lying nearer to the origin is the best one due to ideal requirements and high availability rate.
6.1.1 Scheduling Strategy

The scheduling strategy proposed here considers requirements of a job and resource capabilities of the site, based on this it computes the best matching site for a job. It also includes the common inter-resource dependencies that affect the efficient execution of jobs, including I/O dependence and communication overheads in its decision making process. This allows jobs to be executed competently when allocated to resources that is located at different geographic sites. Job request and site representation of CPU resources is done in terms of MIPS (Million Instruction per Second). Changes in unit representations will not affect the strategy as the aggregation algorithm will result in dimensionless indices as long as the request and site resource representation units are the same. This applies to all other resources shared within a strategy. Scheduling strategy also tries to allocate resources such as to satisfy a job’s requirements in a single site in order to improve performance. It additionally avoids over allocation of resources, so as to prevent the detrimental effects on other jobs which might need these resources to achieve efficiency in execution.

6.1.2 Model of the Grid Environment

The grid environment consists of diverse machine types, disks and networks. Resources in the grid environment are made of desktops, servers, clusters and multi-processor systems. All computation resources are connected through different bandwidths. The network bandwidth connectivity is illustrated in the Figure 6.1. Every resource is accessible to every other participating node in the grid. Changes in any shared resource at a site are known to all locations instantaneously throughout the grid environment. The importance of the resources with respect to each other is identical. Different amounts of CPU MIPS, Hard disk space and Bandwidth are provided by the resources. A user from any node within the grid environment can submit a
job. The execution results of the job are given back to the user. The grid scheduler determines how jobs should be scheduled and how the resources should be utilized. The CPU resource’s computational capability is represented in the form of MIPS. This measure is used to standardize the performance of different CPU architectures in different sites.

![Network bandwidth connectivity](image)

**Figure 6.1 Network bandwidth connectivity**

### 6.1.3 Assumptions in the Execution Environment

1. During execution, the resource requirements for a job do not change.

2. The provided resource requirements for a job are the upper bound of resource usage.
3. Every job has a data requirement in addition to computational requirement. In the data requirement, the data source and size is stated.

4. The data source for a job can be located anywhere in the grid environment.

5. A job can be scheduled anywhere in the grid environment for its execution.

6. Once the distribution of resources starts, the resources are locked for a job’s execution and reclaimed after use.

7. The runtime of a job is calculated from the time the job is submitted till the end of its execution.

8. The time taken for inter-process communication of parallel applications is also included.

6.2 SCHEDULING DIMENSIONS AND INDICES CALCULATION

The dimensions considered for scheduling a job in the proposed algorithm are Computation Data, Deadline and Budget. Computation and Data are the resource requirement classification used to verify the effectiveness of the scheduling strategy. These two dimensions are used to achieve faster computation through proper resource allocation. In the proposed simulation, the resources considered for computational dimension are MIPS (C) and Disk Space (S). Inter resource communication is addressed by the concept of Resource Potential (P). The available resources are aggregated and then combined into two major indices. These two indices are referred as the Computational and Data Index, respectively. Index calculations are similar to the one done in Khoo et al (2007). Deadline and budget are the two new parameters introduced here which is generally
specified by the user within which a job must be executed. A scheduler must select a resource such that it has the capability of satisfying resource requirements of a job and also it must execute within the specified deadline and budget. A 2-D virtual map is constructed for computation and data index. The resources that satisfy both deadline and budget are only considered for virtual map construction. The most suited resource providers will be the sites located nearest to the origin. The following sections will demonstrate how the four selected dimensions are constructed and the process of aggregation that leads to the final aggregated Indices used in the virtual map.

6.2.1 Computation Dimension and Index Calculation

Resources in the computation dimension consist of entities that would impact the efficient computation of a job. Each resource is in turn represented by a capability value and a requirement value. The following allocable resources are used as basis for scheduling in the computation dimension: CPU MIPS (C) and Hard disk space (S).

However, it is noted that this is insufficient to represent a collection of sites and how they can possibly inter-operate with each other. A job submitted to a poorly connected site will be penalized when job fragmentation occurs or when the data required for processing is located in another location. In order to minimize the detrimental effects in such cases, a parameter Resource Potential (P) is used. This parameter assists in the evaluation of the Computation Index.

Assume ‘m’ as the total number of sites in a Grid computing Environment. The potential ‘P’ quantifies the level of network connectivity between itself and neighbouring sites. In the proposed work, network latencies and communication overhead are assumed to be inversely
proportional to bandwidth. Resource potential $P_i$ of a resource $R_i$ is referred in the form of ‘virtual distance’ where $1 \leq i \leq m$. This is calculated as $P_i = \sum B_{ij}$, where $B$ is the upload bandwidth expressed in bits per second for $R_i$ to $R_j$ for $i \neq j$ and $B_{ij} = 0$ if $i = j$. This flattens the bandwidth view of all the resources to the maximum achievable bandwidth between resources and hence eliminates network complexities.

In a grid environment, resources are described as a set $R = \{R_1, \ldots, R_m\}$, allocable computation resource within a site $l$ as a set $S_d = \{R_i, t\}$ where $S_d$ is a subset of $R$ and $t$ denotes time. $R_i$ is represented by a 3 tuple of $f_i(\langle C, S, P >, t)$ denoting the dimensions considered. Job environment is described as $J = \{A_1, \ldots, A_l\}$ and computation requirements of each job $A_j$ in the set of $J$ jobs is represented by $g_j(\langle C, N, P_{src} >, t)$. The values of $C$, $S$ and $P$ dynamically change with resource availability over time $t$.

Evaluation of various resource requirements of sites and jobs allows to aggregate their values and encoding inter-resource relationships in order to arrive at a single computational index such that it can be used to obtain the allocation score. This is done by obtaining a ratio of provision ($R_{ij}$), for site $i$ and job $j$, between what is requested and what is possibly provided. For computational resources, it is given by,

$$R_{ij} \{C\} = 1 - \frac{f_i \{C\}}{g_j \{C\}} \quad (6.1)$$

where,
- $f_i \{C\}$ is MIPS resource provided at site $i$
- $g_j \{C\}$ is MIPS resource required by job $j$

Only positive values are considered for drawing the virtual map and so the values for $R_{ij}$ are truncated to 0 if the ratio value evaluates to be less than 0.
The following observations are made in the calculation of ratio of provision:

a. If the value is 0, perfect ability to provision for a resource.

b. If the value is >0, inability to provide for a resource.

c. If the value is <0, over ability to provision resources, Set the value as 0.

The same ratio is applied to all the resources and resource requirements in the computational index, which also includes Hard disk (F) requirements. The ratio between the potential value of the site \( P_i \) and the source file potential \( P_{src} \) are also included. This evaluates whether site connectivity is equal or better to where the source data file is located. To merge all the provisioning ratios to a single dimensionless computation index, Euclidean distance between the provisioning ratios is calculated.

In general, Euclidean distance between \((x_1,y_1,z_1)\) and \((x_2,y_2,z_2)\) is given by,

\[
\text{Euclidean distance} = \sqrt{[(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2]} \quad (6.2)
\]

In the construction of virtual map for each job, the Euclidean distance is computed between the origin and each of the provisioning ratios. These ratios are then aggregated to a dimensionless computational index \(x_{ij}\) as

\[
x_{ij} = \sqrt{((R_{ij\{C}\} - 0)^2 + (R_{ij\{S}\} - 0)^2 + (R_{ij\{P}\} - 0)^2)} \quad (6.3)
\]

Hence the computation index evaluates to

\[
x_{ij} = \sqrt{(R_{ij\{C\}})^2 + R_{ij\{S\}}^2 + R_{ij\{P\}}^2} \quad (6.4)
\]
6.2.2 Data Dimension and Indexing

In the data dimension, the I/O of a job is affected by the interrelated resources and an index is evaluated that aids in determining a good resource site that would best execute a job. The expected time for I/O is determined based on the estimated data communications required and the bandwidth between the source file location and the target job allocation site. The ratio between the I/O communication time to the estimated local job runtime is then taken. This ratio allows to evaluate the level of advantage a job has in dispatching that job to a remote site. Thus, allocation of a job to the intended target resource should be one whereby this ratio is as low as possible. The I/O time is mainly dependent on the availability of bandwidth at a site.

The bandwidth $B$ between two sites $i$ and $j$ is annotated as $B_{ij} = \min\{B_{ij}^{\text{download}}, B_{ji}^{\text{upload}}\}$ which changes over time $t$ as data capabilities of a resource $S_d \{ R_i, t \}$ where each item in the set is represented by $d_i \{ \langle B \rangle, t \}$. The data requirement of a job $j$ is thus represented by $e_j \{ \langle F, A_{\text{runtime}} \rangle, t \}$ where $A_{\text{runtime}}$ is the estimated runtime of the job. Data index ($y_{ij}$) is given by,

$$y_{ij} = e_j \{ F \}/ (d_i \{ B_{ij} \} . A_{\text{runtime}} )$$

(6.5)

The evaluation is an example of aggregation based on resource inter-relation. I/O time is affected by the amount of data for a job and the actual bandwidth resource available. In the worst-case scenario, the amount of data required for the job would also be the amount of hard disk resource required at the site to store the data to be processed. This therefore interrelates the data resources to the bandwidth resources available.

It is noted that $y_{ij}$ continues to be dimensionless and a smaller value would represent a better site $i$ preference when compared to a larger one. An
(ascending) ordered \( y_{ij} \) would rank sites with the better advantage in handling job fragmentation compared to those ranked later.

### 6.2.3 Deadline and Indexing

Deadline, the new dimension proposed here describes a computational economy framework for regulating the supply and demand for resources. It allocates the resources to a job based on the users’ quality of services requirements. Calculation of deadline optimizes for time and thereby helps to achieve a lower job completion time. The deadline is also computed at each site in order to find the best resource at a site which efficiently executes a job within the user specified deadline \((U_d)\).

The deadline \( D_{ij} \) for job \( j \) in a site \( i \) is computed as follows:

The runtime of each job \( (A_{\text{runtime}}) \) is estimated by,

\[
A_{\text{runtime}} = \frac{\text{job length}}{\text{resource (MIPS)}} \quad (6.6)
\]

Thus deadline index is given by,

\[
D_{ij} = (1 - (A_{\text{runtime}} / U_d)) \quad (6.7)
\]

If the ratio for deadline

a. lies between 0 and 1, the resource is suitable for job allocation.

b. is greater than 1, the resource is not suitable for job allocation.

### 6.2.4 Budget

As the grid environment is heterogeneous in nature, each resource may have different configurations. The resource rate varies from one another. According to job requirements, cost of each resource in executing a job is
calculated. Then the resource cost \( (CO_i) \) is compared with the user specified budget\( (BU_j) \). If both match, the job is submitted to an appropriate resource.

The budget index \( (BI_{ij}) \) is given by,

\[
BI_{ij} = \frac{CO_i}{BU_j}
\]  \hspace{1cm} (6.8)

where, \( CO_i = Pr_i \times A_{runtime} \) \hspace{1cm} (6.9)

\( Pr_i \) is the Cost of the resource at unit time.

6.3 PRO-ACTIVE FAILURE HANDLING STRATEGY

Pro-active failure handling method improves the scheduling algorithm by being able to prevent job failures during execution. Inability to account for failure during allocation will still cause a slow-down in job completion time if it is to occur in the midst of job execution. This could be avoided if the system is able to detect it priorly. The scheduling algorithm is modified so as to avoid job failures upon scheduling and thus capable of improving the job reliability. This is in contrast with passive failure handling (Lee et al 2004 and Frey et al 2001) where the handling of failures by scheduling algorithm occurs after the allocation of resources. Figure 6.2 (a) and Figure 6.2 (b) shows the passive and pro-active failure handling methods.

The stages of availability of resources can be described as,

2. Resource continues to be available as none of the components within itself has failed.
3. Resource encounters a failure in one of its components and goes offline for maintenance and fix.
4. Resource goes through a series of checks, replacements or restarts to see if it is capable to re-join the Grid Computing Environment.

5. Resource comes online and becomes available to the Grid Computing Environment (return to first stage).

From the above, it was observed that in stages (2) and (4), the resource undergoes a period of uncertainty. This uncertainty stems from the fact that the resource probably might not fail or recover for a certain period of time.

![Diagram](a) Passive Method

![Diagram](b) Pro-Active Method of Failure Handling

Figure 6.2 (a) Passive Method (b) Pro-Active Method of Failure Handling
In pro-active mechanism (Khoo and Veeravalli 2010), the failure consideration for the grid is made before the scheduling of a job (Figure 6.2(b)), and dispatched with the expectation that the job does not fail. This is in contrast with passive mechanism, (Litzkow et al 1988), where the algorithm handles job failures after they have occurred. Pro-active failure handling strategies allow resource scheduling algorithm to be modified to avoid job failures upon scheduling.

### 6.3.1 Mathematical Modeling

In order to construct a pro-active scheduling strategy, a mathematical model is used to determine the availability of resources. The model is constructed to predict the capacity in a Grid Computing Environment (GCE) given a total fixed number of resources that can possibly participate in the environment. The mathematical model is based on distribution function using Mean Time to Failure, Mean Time to Recovery and reliability values of each resource.

The purpose of the mathematical model is to Estimate the number of nodes in a Grid at a certain time and to calculate the probability of a job being able to complete its execution. Addressing these two criteria will allow the strategy to dispatch jobs only to resources that will more likely guarantee the successful completion of the job, and know ahead the likely capacity of the GCE at a point in the future. A new dimension called availability index is calculated based on the following variables:

- **MTTF and \( \lambda_F \)**: The Mean Time to Failure represents the average amount of time a resource is available to the GCE before going offline. The average rate of failure is termed as \( \lambda_F = 1/\text{MTTF} \).
137

- MTTR and $\lambda_R$: The Mean Time to Recovery represents the average amount of time taken for a resource to rejoin the GCE after going offline. The average rate of recovery is termed as $\lambda_R = 1/\text{MTTR}$.

\[
\text{Availability Index (AI)} \ z_{ij} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})} \quad (6.10)
\]

6.4 CONSTRUCTION OF VIRTUAL MAP

From the Computation, Data, Deadline, Budget and Availability index calculated for each resource, the best resource for job submission is identified as one that has low index values. Computation, Data and Availability indices are used to construct a 3D plot which is termed as the Virtual map. Only the sites that satisfy both deadline and budget are included in the virtual map. The site which position itself closest to the origin is the best suited resource for a job as it deviates from the resource requirements by the least amount. The Euclidean distance from the origin denotes the best resources that match the job’s resource requirements for a time instance. A Virtual map is drawn for each job as the job requirements differ for each submitted job. Hence the map should be drawn every time when the job is submitted.

In the simulation, five sites of different configurations are considered. Each site is connected to one another through different bandwidths (Mbps). The capacity for computation in a CPU resource is provided in the form of MIPS. The capacity for storage in each site is expressed in GB. The configuration of the five sites is listed in the Table 6.1.
Table 6.1 Sample Site Configuration

<table>
<thead>
<tr>
<th>Site</th>
<th>CPU (MIPS)</th>
<th>HDD (GB)</th>
<th>Potential (Mbps)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1885</td>
<td>100</td>
<td>103</td>
</tr>
<tr>
<td>B</td>
<td>4270</td>
<td>200</td>
<td>103</td>
</tr>
<tr>
<td>C</td>
<td>7155</td>
<td>300</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>10540</td>
<td>400</td>
<td>106</td>
</tr>
<tr>
<td>E</td>
<td>14425</td>
<td>500</td>
<td>106</td>
</tr>
</tbody>
</table>

* Potential is calculated as $\sum B_{ij}$, where i is the site and j is the job.

6.4.1 Indices Calculation

For each resource in the Grid Environment, Computation, Data, Deadline, Budget and Availability indices are calculated. A 3D virtual map is plotted for Computation, Data and Availability indices. Resources are allowed to plot in the virtual map only if it has deadline and budget values less than 1. The resource which is lying nearer to the origin is the best resource for job submission.

6.4.2 Virtual Map for 4D and 5D Scheduling Algorithms

An enhanced multi-dimensional scheduling algorithm is implemented with and without failure handling using Gridsim simulator. The result of the simulation is shown in Figure 6.3 and Figure 6.4. For each resource in the environment, computation index, data index, deadline, budget and availability index are calculated. Figure 6.3 shows a 2D plot with Computation and Data index without failure handling methods. Only the resources that have the capability of executing within the user specified deadline and budget are allowed to plot in a virtual map. Figure 6.4 shows the
3D plot that includes Availability Index \((z_{ij})\) as third dimension along with the dimensions of 2D plot. Availability index gives the probability of each resource in the UP state. The resource that is lying nearer to the origin in the virtual map is the best resource for job execution.

**FIGURE 6.3  2D Virtual Map for Job 1**

As requirements differ for each job, the Virtual Map is essentially different for each job submitted. Indices have to be computed for each time a job is being submitted or re-submitted in the Grid Computing Environment.
For the same set of job specification, Figure 6.3 shows resource B as suitable resource for job execution, whereas Figure 6.4 (scheduling algorithm with failure handling) shows resource D as suitable resource. Even though both resource B and D are capable of executing a job, Resource D has the highest probability of execution without failure. Probability of resource in UP state at time t+T calculated based on its MTTR and MTTF values assist in selecting the suitable resource. The probability of Resource D in UP state is higher than the probability of Resource B, hence D is chosen as a suitable resource.

### 6.5 EVALUATION OF THE PROPOSED SCHEDULING ALGORITHMS

The workload model of Song et al (2005) is used as the input to the simulation model. In the simulation model, jobs are allowed to arrive in a stream over a span of 24 hours. Job requirements are modeled and fed to the
simulation model. Scheduling algorithm allocates a job to a suitable resource based on various allocation policies. The four dimensional scheduling algorithm developed without incorporating failure handling strategy is compared with backfill algorithm based on the performance metrics, Average Waiting Time (AWT) and Queue Completion Time (QCT).

AWT is a measure of responsiveness of the scheduling mechanism. A low wait time suggests that the algorithm can potentially be used to schedule increasingly interactive applications due to reduced latency before a job begins execution. QCT, when coupled with the average waiting time of a job, allows us to deduce the maximum amount of time a typical job will spend in the system for a given workload.

**Average Wait-Time (AWT)**

AWT is defined as the time duration for which a job waits in the queue before being executed. The wait time of a single job instance is obtained by taking the difference between the time the job begins execution ($e_j$) and the time the job is submitted ($s_j$). This is computed for all jobs in the simulation environment. The average job waiting time is then obtained. If there are a total of $J$ jobs submitted to a GCE, the AWT of a job is given by,

$$AWT = \frac{\sum_{j=0}^{j-1} (e_j - s_j)}{J} \quad \text{(6.11)}$$

**Queue Completion Time (QCT)**

QCT is defined as the amount of time it takes for the scheduling algorithm to be able to process all the jobs in the queue. This is computed by tracking the time when the first job enters the scheduler until the time the last
job exits the scheduler. In the experiment done, the number of jobs entering the system is fixed, to make the simulation more traceable. This allows the quantitative measure of throughput, where the smaller the time value, the better. The queue completion time is given by,

$$QCT = e_j + E_J - s_0 \quad \text{(6.12)}$$

where, $E_J$ is the execution time of the last job. This includes the I/O and communication overheads that occur during job execution.

6.5.1 Comparison of 4D Scheduling Algorithm with Backfill Algorithm

The four dimensional scheduling algorithm uses Computation, Data, Deadline and Budget as dimensions for scheduling. The proposed 4D algorithm is compared with the conventional backfilling strategy which is similar to the one used in Korpela et al (2001). The Average waiting time and Queue Completion Time are the parameters used to study the performance of the algorithm. Backfill algorithm allows a scheduler to make use of available resources by running jobs out of order. During backfilling, few higher priority jobs may get delayed due to smaller jobs. This tends to rise in average waiting time and Queue Completion Time. The proposed 4D scheduling algorithm is compared with Backfill Algorithm (Hamscher et al 2000 and Mu’alem and Feitelson 2001) which is shown in Figure 6.5; the graph shows that the Average waiting Time and Queue Completion Time of the proposed scheduling algorithm is reduced when compared to the Backfill algorithm.
Figure 6.5 Normalized Performance of the proposed 4D scheduling algorithm against Backfill Algorithm

The proposed 4D Scheduling algorithm has reduced the AWT and QCT by 28.6% and 28.6% respectively when compared with the traditional Backfill algorithm. As the proposed 4D scheduling algorithm performs better than the backfill algorithm, proactive failure handling strategies are included in the proposed algorithm for further enhancement.

6.5.2 Comparison of the Proposed 5D Scheduling Algorithm with Backfill and 4D Scheduling Algorithm

The proposed 4D scheduling algorithm (without failure handling) and Backfill algorithm are compared with the proposed 5D Scheduling algorithm (with failure handling) using the metrics- Job Processing Rate, Job Rejection Rate and Job Failure Rate. However, the measures AWT and QCT are not suited for investigating the effectiveness in the event of faults in the grid environment, the effectiveness of 5D scheduling algorithm is evaluated by capturing the job failure and rejection rates in each simulation. A job is defined to have failed when its execution is terminated due to a resource
failure. A job is rejected when its resource request exceeds what is stated available in the scheduling algorithm. The job processing rate is also captured as an indication of throughput of the resulting algorithm.

In order to measure the performance of scheduling algorithm with and without failure handling mechanisms, the following metrics are used.

Job Processing rate [JPR]

\[
JPR = \frac{\text{Number of jobs successfully completed}}{\text{Total Queue Completion Time}}
\]  

(6.12)

A higher JPR will indicate a larger number of successfully completed jobs or a lower queue completion time. A high JPR will therefore indicate that an algorithm is capable of high throughput.

Job Failure rate [JFR]

\[
JFR = \frac{\text{Number of jobs failed at runtime}}{\text{Total Queue Completion Time}}
\]  

(6.13)

A low JFR is desired as it signifies the reduction in the number of job failures during the course of its queue completion. This indicates that a strategy that is able to allocate resources will reduce the number of jobs failing in its course of execution.

Job Rejection Rate [JRR]

\[
JRR = \frac{\text{Number of jobs rejected}}{\text{Total Queue Completion Time}}
\]  

(6.14)
A job is rejected when its resource request exceeds what is stated available in the scheduling algorithm. A low JRR indicates the ability of an algorithm to handle all types of jobs submitted to the queue.

Table 6.2 lists the parameter values calculated for different algorithms considered. Figure 6.6 illustrates that the scheduling algorithm with failure handling improves the job processing rate by 2.4 times and reduces the job failure rate by 39.8% when compared to scheduling algorithm without failure handing. Job rejection rate of 5D algorithm is greater than 4D algorithm because 5D algorithm predicts the resources in the grid environment before execution. An algorithm rejects job in cases when Grid environment does not contain enough reliable resources or if job’s resource requisition exceeds what is stated available. As both 4D and 5D use same allocation principle, the rejection rate in rejecting jobs due to insufficient resources is same and meanwhile it is lower in 4D algorithm when considering reliability into account. As rejection rate in 4D Scheduling algorithm is low, the failure rate is high. In 5D scheduling algorithm where failure handling is included, though the rejection rate is comparatively more than other algorithms, job failure rate is very low and all the scheduled jobs are completed without failure.

Table 6.2 Comparison of the proposed algorithms with respect to job Processing, Failure and Rejection

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Job Processing Rate</th>
<th>Job Failure Rate</th>
<th>Job Rejection Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed 5D Scheduling Algorithm (With failure handling)</td>
<td>41.22</td>
<td>20.61</td>
<td>19.23</td>
</tr>
<tr>
<td>Proposed 4D Scheduling Algorithm (Without failure handling)</td>
<td>17.13</td>
<td>34.26</td>
<td>17.13</td>
</tr>
<tr>
<td>Backfilling Algorithm</td>
<td>15.2</td>
<td>35.2</td>
<td>16.8</td>
</tr>
</tbody>
</table>
Thus in the present work, a new scheduling algorithm that satisfies resource requirements as well as economic parameters for a job that arrive in a grid system has been considered. This enhanced scheduling algorithm allows the user to specify deadline and budget along with resource requirements within which a job must be executed. Job and resource relationships are captured and are used to create an aggregate index. A 3D virtual map is plotted for Computation, Data and Availability indices. Resources are allowed to be plotted in a virtual map only if it has deadline and budget value less than 1. The resource which is lying nearer to the origin is the best suited resource for job submission. The proposed 4D Scheduling algorithm has reduced the AWT and QCT by 28.6% and 28.6% respectively when compared with the traditional Backfill algorithm.

Pro-active failure handling method is also included in the scheduling algorithm which estimates the availability of grid resources and
avoids rescheduling of jobs during execution time. The simulated results show that the proposed scheduling algorithm gives better result when compared with the existing algorithms. In order to get the values for metrics, a series of jobs are made to run in the Grid Environment. The job processing rate, job failure rate and job rejection rate are calculated for scheduling algorithm with failure handling and without failure handling. Results show that the scheduling algorithm with failure handling improves the job processing rate by 2.4 times and reduces the job failure rate by 39.8% when compared to scheduling algorithm without failure handling.