Research Journal of Applied Sciences 6 (7-12): 525-531, 2011
ISSN: 1815-932X
© Medwell Journals, 2011

# Job Scheduling with Economic Parameters and Failure Handling Methods in a Grid Computing Environment

[1]L.M. Nithya and [2]A. Shanmugam
[1]Department of Information Technology, SNS College of Technology,
Coimbatore, Tamil Nadu, India
[2]Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, India

**Abstract:** In this study, it is proposed to enhance the grid job scheduling algorithm by including economic parameters-deadline and budget that schedule the job to a suitable resource which is capable of satisfying not only computational requirements but also economic constraints of a job. As grid environment is heterogeneous in nature, there may be wide range of failures that might affect job execution. In order to handle failures and to avoid rescheduling of jobs after the detection of failure, pro-active failure handling methods are included in the enhanced scheduling algorithm to estimate the availability of resources in the grid and also to preemptively calculate the expected long term capacity of the grid. The performance indicators such as average waiting time and queue completion time are calculated for the proposed job scheduling algorithm and backfill algorithm, comparison exemplifies that the proposed algorithm gives significant results. The proposed scheduling algorithm with failure handling methods outperforms the scheduling algorithm without failure handing in most of the situations. It has increased the job processing rate by 58%, decreased the job failure rate and job rejection rate by 66 and 16%, respectively.

**Key words:** Grid job scheduling, pro-active failure handling, economic parameters, environment, algorithm, task

## INTRODUCTION

Grid computing is characterized by large-scale sharing cooperation of dynamically distributed resources such as CPU cycles, communication bandwidth and data to constitute a computational environment. The concept of grid computing has pushed the envelope of distributed computing by moving the local resources such as memory, disk and CPUs to a wide area distributed computing platform sharing these very same resources (Foster and Kesselman, 2004; Foster *et al*., 2001). Grid middleware systems are aimed to perform various tasks such as collecting information about participants, bringing different requirements to common standards, carrying negotiations, monitoring queues, security issues, etc. Among the others, one of the most important tasks of any grid coordinator is an effective allocation of jobs to available resources. This is usually done by grid scheduling systems. Architecture of a common grid scheduler is shown in Fig. 1. Grid scheduling (in general case) is known to be hard to solve NP-complete problem. Grid resources are used for solving various kinds of large scale parallel applications in physics, engineering and commerce. These applications may be submitted by grid users via grid middleware and may have tight budget and deadline constraints. Running applications in such an environment is susceptible to wide range of failures as revealed by a survey Medeiros *et al*. (2003) with real users on fault treatments in the grid.

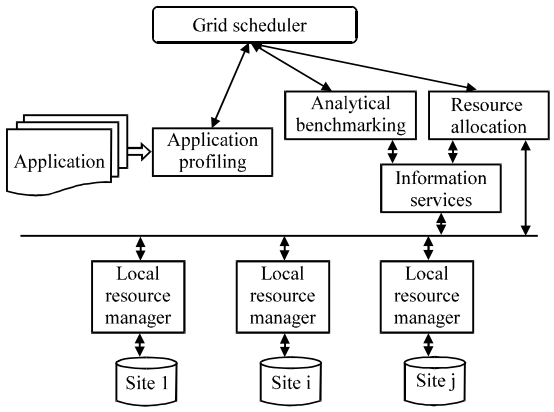


Fig. 1: Architecture of a common grid scheduler

Given the ability to preemptively know about failures and to handle them adequately would allow the scheduling algorithms to prevent job. Handling job

**Corresponding Author:** L.M. Nithya, Department of Information Technology, SNS College of Technology, Coimbatore, 641 035 Tamil Nadu, India

failures can help to reduce the turn-around time for successful job completion, it would then be possible to create large scale scheduling algorithms where it is able to effectively estimate and allocate jobs to resources that can fulfill their task with minimal interruptions and re-scheduling. This will ultimately result in higher throughput and a higher level of quality for jobs submitted to grids.

**Related work:** Researchers have investigated grid scheduling by optimization in the computation domain or in data or I/O domain. Job optimization is achieved by sending the jobs to multiple sites instead of to the least loaded site (Subramani and Srinivasanet, 2002). The algorithm in Ranganathan and Foster (2002) handles data and computation requirements separately. While this may be suitable for independent tasks, it is less effective when applied to inter-resource dependency types of job. The algorithm proposed in Khoo *et al.* (2007) inter-relates both data and computation requirements for scheduling of jobs. Along with the two dimensions in Khoo *et al.* (2007), economic parameters, deadline and budget are included in the present study as new dimensions for scheduling of job which is referred to as 4D scheduling algorithm.

Grid failures can be handled using pro-active or passive mechanisms. By pro-active mechanisms, the failure consideration for the grid can be made before scheduling of a job and dispatched with hopes that the job does not fail. Passive mechanisms identify algorithms that handle the job failures after they have occurred. Many researches are passive in nature and deal with failures through grid monitoring as mentioned in (Subramani and Srinivasanet, 2002; Kang and Grimshaw, 2007). These methods mainly do so by monitoring for failures followed by either checkpoint-resume or terminate restart (Lee *et al.*, 2004; Litzkow *et al.*, 1988; Frey *et al.*, 2001; Ayyub *et al.*, 2007). Replication (Li and Mascagni, 2003) is one of the passive failure mechanisms where a task is replicated on multiple machines in the assumption that at least one copy of the task will execute successfully. This can possibly lead to an over allocation of resources which will be reflected as an opportunity cost on other jobs in the execution queue.

Pro-active strategy addressed in Khoo and Veeravalli (2010) is incorporated into the proposed algorithm hence it handles failure before it occurs by analysis of individual node failure predictions. This further improves the performance of the proposed 4D scheduling algorithm which is henceforth referred as 5D scheduling algorithm.

## MATERIALS AND METHODS

**Scheduling strategy:** In grid computing because 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 access. 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 behaviors of resources and is able to assign tasks to resources according to certain performance requirements.

The scheduling strategy proposed here considers requirements of a job and resource capabilities of the site and based on this it computes the best matching site for a job. It also includes the common inter-resource dependencies that affects 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 as an indication of performance. Future changes in unit representations will not affect the strategy as the aggregation algorithm will result in dimensionless indexes 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.

**Selection of scheduling dimensions:** 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 by Khoo *et al.* (2007). Deadline and budget are the 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 study will demonstrate how we construct the four selected dimensions and the process of aggregation that leads to the final aggregated indexes used in the virtual map.

**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. Researchers make use of the following allocable resources 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 is used. This is to assist in the evaluation of the computation index.

Evaluation of various resource requirements of sites and jobs allows us 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:

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

Only the positive values of $R_{ij}\{C\}$ are considered such that $R_{ij}\{C\} = 0$ if the above evaluates to be less than zero. $f_i\{C\}$ and $g_j\{C\}$ are the MIPS resource provided at site i and MIPS resource required by job j. Only positive values are considered in the virtual map.

The same ratio of provision is applied to all resource and requirements within the computational dimension. Additionally the ratio of provision are included between the potential value of the site ($P_i$) and the source file potential ($P_{src}$). This allows us to evaluate if site connectivity is equal or better to where the source data file is located. This ensures that the possible target job submission site will not be penalized more than required if job fragmentation is to occur when compared to executing the job in place at the data source location. These ratios are then aggregated into a dimensionless computation index ($x_i$) for site i and job j using Eq. 2:

$$x_{ij} = \sqrt{((R_{ij}\{C\}^2 + R_{ij}\{S\}^2 + R_{ij}\{P\}^2)} \quad (2)$$

**Data dimension and indexing through resource inter-relation:** In the data dimension, the I/O of a job is affected by the inter-related resources and an index is evaluated that aids us 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 us 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 $B_{ij} = \min\{B_{ij}^{download}, B_{ji}^{upload}\}$ 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 $d_i\{<B>, t\}$. The data requirement of a job j is thus represented by $e_j\{<F, A^{runtime}>, t\}$ where $A^{runtime}$ is the estimated runtime of the job. Data index ($y_{ij}$) is given by:

$$y_{ij} = e_j\{F\}/(d_i\{B_{ij}\}.A^{runtime}) \quad (3)$$

This evaluation is an example of aggregation based on resource interrelation. 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, inter-relates 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.

**Deadline:** 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 ($A^{runtime}$) is estimated by:

$$A^{runtime} = \text{job length/resource MIPS} \quad (4)$$

Thus, deadline index is given by:

$$D_{ij} = (1-(A^{runtime}/U_d)) \quad (5)$$

If the ratio for deadline lies between 0 and 1, the resource is suitable for job allocation if it is >1, the resource is not suitable for job allocation.

**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 is compared with the user specified budget. If both matches, the job is submitted to an appropriate resource.

**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 made aware of it.

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; Frey *et al*., 2001)) where the handling of failures by scheduling algorithm occurs after the allocation of resources. Figure 2 shows the steps of pro-active failure handling method.

**Estimating resource availability through mathematical modeling:** The stages of availability of resources can be defined as:

- Resource available to the grid computing environment
- Resource continues to be available pending that none of the components within itself has failed
- Resource encounters a failure in one of its components and goes offline for maintenance and fix
- Resource goes through a series of checks, replacements or restarts to see if it is capable to re-join the grid computing environment
- Resource comes online and becomes available to the grid computing environment (return to first stage)

From the above, it was observed that in stages and 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.

A mathematical model is constructed to predict the capacity in a GCE given a total fixed number of resources that can possibly participate in the environment. The mathematical model is based on Poisson's exponential 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 important points 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. Researchers also term the average rate of failure to be $\lambda_F = 1/MTTF$.

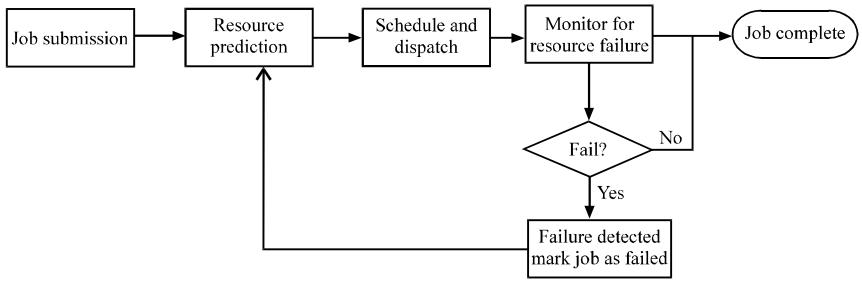


Fig. 2: Pro-active failure handling method

**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. Researchers also term the average rate of recovery to be $\lambda_R = 1/MTTR$.

$$\text{Availability Index (AI) } z_{ij} = \Sigma MTTF/(\Sigma MTTF+\Sigma MTTR) \quad (6)$$

**Construction of 3D virtual map:** The 2D virtual map is extended to 3D virtual map by including availability index as third dimension. This availability index ranges from 0-1 and corresponds directly to the probability of each resource in the UP state. Resource selection is now based on the minimum Euclidean distance to the origin based on the values provided by all the three axes. This allows us to consider factors such as computation, data as well as availability provided by that of GCE resource with only a linear increase in computational complexity of the allocation strategy.

**Performance indicators:** In order to measure the performance of the proposed 4D scheduling algorithm, researchers use the metrics Average Wait 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.

However, as these measures are not suited for investigating the effectiveness in event of faults in the grid environment, researchers evaluate the effectiveness of 5D scheduling algorithm 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.

## RESULTS AND DISCUSSION

Researchers have implemented an enhanced multi-dimensional scheduling algorithm with and without failure handling using gridsim simulator. The result of the simulation is shown in Fig. 3 and 4. For each resource in the environment, computation index, data index, deadline, budget and availability index are calculated. Figure 3 shows a 2D plot with computation and data index without failure handling methods. Only the resources that have

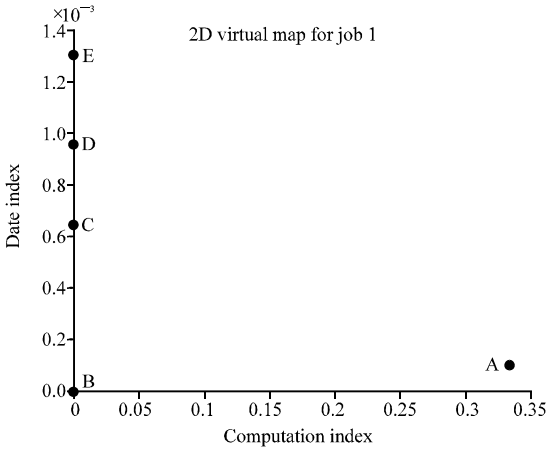


Fig. 3: 2D virtual map including economic parameters (without failure handling)

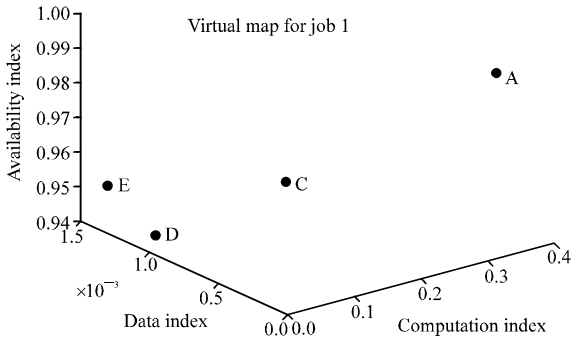


Fig. 4: 3D virtual map including economic parameters (with failure handling)

the capability of executing within the user specified deadline and budget are allowed to plot in a virtual map. Figure 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.

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 a same set of job specification, Fig. 3 shows resource B as suitable resource for job execution whereas Fig. 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. The proposed algorithm uses computation, data, deadline and budget as dimensions for scheduling.

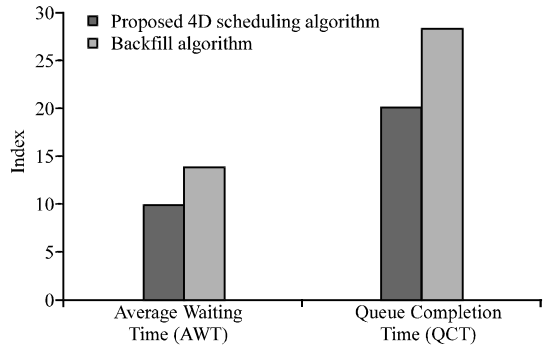


Fig. 5: Performance analysis of the proposed 4D scheduling algorithm with backfill algorithm

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 which is shown in Fig. 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.

The proposed 4D scheduling algorithm has reduced the AWT and QCT by 28 and 29%, 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.

**Performance evaluation:** The proposed 4D scheduling algorithm (without failure handling) is compared with the proposed 5D Scheduling algorithm (with failure handling) using the metrics-job processing rate, job rejection rate and job failure rate. Figure 6 shows that the scheduling algorithm with failure handling outperforms scheduling algorithm without failure handing by 58.4% in job processing rate and 66.22% in job failure rate. Job rejection rate of 5D algorithm is almost similar or >4D algorithm because 5D algorithm predicts all 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 4 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

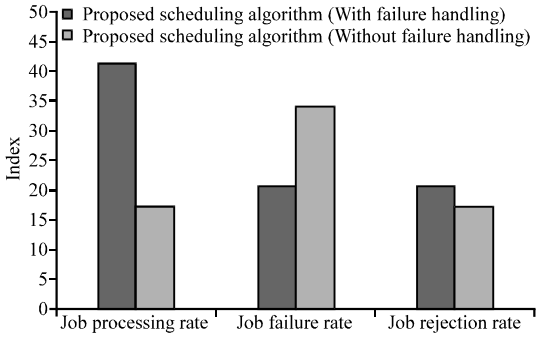


Fig. 6: Performance analysis of the scheduling algorithm with and without failure handling methods

algorithm when considering reliability into account. As rejection rate in 4D scheduling algorithm is low, the failure rate is increased considerably.

## CONCLUSION

In this study, a 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 the resource requirements within which a job must be executed. 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 future the best passive methods can be used along with the pro-active methods to further address failures that occur during runtime. The number of dimensions can be extended by including quality-of-service parameters for scheduling.

## REFERENCES

Ayyub, S., D. Abramson, C. Enticott, S. Garic and J. Tan, 2007. Executing large parameter sweep applications on a multi-VO testbed. Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, January 24, 2007, Los Alamitos, CA., USA., pp: 73-80.

Foster, I. and C. Kesselman, 2004. The Grid: Blueprint for a New Computing Infrastructure. 2nd Edn., Morgan-Kaufman, Los Altos, CA., USA., ISBN: 9781558609334, Pages: 748.

Foster, I., C. Kesselman and S. Tueke, 2001. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. Supercomput. Appl., Vol. 15 10.1.1.24.9069.

Frey, J., T. Tannenbaum, I. Foster, M. Livny and S. Tuecke, 2001. Condor-G: A computation management agent for multi-institutional grids. Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, August 7-9, 2001, San Francisco, CA., USA., pp: 55-63.

Kang, A.W. and A. Grimshaw, 2007. Grimshaw: Failure prediction in computational grids. Proceedings of the 40th Annual Simulation Symposium, March 26-28, 2007, Norfolk, VA., USA., pp: 275-282.

Khoo, B.T.B, B. Veeravalli, H. Terence and S.C.W. Simon, 2007. A multi-dimensional scheduling scheme in a grid computing environment. J. Parallel Distrib. Comput., 67: 659-673.

Khoo, B.T.B. and B. Veeravalli, 2010. Pro-active failure handling mechanisms for scheduling in grid computing Environments. J. Parallel Distrib. Comput., 70: 189-200.

Lee, H.M., S.H. Chin, J.H. Lee, D.W. Lee, K.S. Chung, S.Y. Jung and H.C. Yu, 2004. A resource manager for optimal resource selection and fault tolerance service in grids. Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, April 19-22, 2004, Chicago, IL., USA., pp: 572-579.

Li, Y. and M. Mascagni, 2003. Improving performance via computational replication on a large-scale computational grid. Proceedings of the 3rd IEEE International Symposium on Cluster Computing and the Grid, May 15-18, 2003, Tokyo, Japan, pp: 442-448.

Litzkow, M., M. Livny and M. Mutka, 1988. Condor: A hunter of idle workstations. Proceedings of the 8th International Conference of Distributed Computing Systems, June 13-17, 1988, San Jose, CA., USA., pp: 104-111.

Medeiros, R., W. Cirne, F Brasileiro and J. Sauve, 2003. Faults in grids: Why are they so bad and what can be done about it?. Proceedings of the 4th International Workshop on Grid Computing, November 17, 2003, Phoenix, AZ., USA., pp: 18-24.

Ranganathan, K. and I. Foster, 2002. Decoupling computation and data scheduling in distributed data-intensive applications. Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, July 24-26, 2002, Edinburgh, Scotland, pp: 352-358.

Subramani, V.K. and R. Srinivasanet, 2002. Distributed job scheduling on computational grids using multiple simultaneous requests. Proceeding of the 11th IEEE International Symposium on High Performance Distributed Computing, 2002, Edinburgh, Scotland, pp: 359-367.