Hierarchical Replication Strategy for Adaptive Scoring Job Scheduling in Grid Computing

S.Gomathi Subbu,
PG Student,
SVS College of Engineering,
Coimbatore, Tamilnadu, India

Abstract
Grid technology, which together a number of personal computer clusters with high speed networks, can reach the same computing power as a supercomputer does, also with a minimum cost. However, heterogeneous system is called as grid. Scheduling independent tasks on grid is more difficult. In order to utilize the power of grid completely, we demand an efficient job scheduling algorithm to execute jobs to resources in a grid. The Data Grid provides massive aggregated computing resources and distributed storage space to deal with data-intensive applications. Due to the limitation of available resources in the grid as well as construction of huge volumes of data, efficient usage of the Grid resources becomes a significant challenge. In previous work develop the Adaptive Scoring Job Scheduling algorithm (ASJS) for the grid environment. In that algorithm is not suitable for replication technique. Data replication is a key optimization technique for reducing access latency and managing large data by storing data in a wise manner. Effective scheduling in the Grid can reduce the amount of data transferred between nodes by submitting a job to a node where most of the requested data files are available. The proposed system uses dynamic data replication strategy, called Effective Hierarchical Replication (EHR) that improves file access time. This strategy is an enhanced version of the Dynamic Hierarchical Replication strategy. It uses an economic model for file deletion when there is not enough space for the replica node. So our proposed system finds the replicate detection of files with different cluster structure representation of the input files. We combine the replica strategy with ASJS algorithm for efficiently decrease the completion time of submitted jobs, which may consist of computing-intensive jobs and data-intensive jobs.

Keywords
ASJS, EHR, Grid, Replication, Scheduling, Job.

1. INTRODUCTION
A large number of scientific applications such as high energy physics, climate change, and earth observation generate huge amounts of data per year. Today, management and efficient use of large distributed resources are important issues in scientific research and commercial applications. The Grid is a solution for this problem. The Grid can be divided into two parts, Computational Grid and Data Grid. Computational Grids are used for computationally intensive applications that require small amounts of data. But, Data Grids deal with the applications that require studying and analyzing massive data sets. Data Grid in a layered architecture. In that grid architecture have four components. The components are fabric, connectivity, services and application layers. The grid fabric layer involves distributed computational resources, storage resources, and instruments that are connected by high bandwidth. The connectivity layer contains protocols used to query resources in the grid fabric layer and to control transferring data between them. The Data Grid service layer consists of several services such as replication, data discovery and resource brokering. The application layer contains the user applications that work within a virtual organization environment.

The size of data that are requested in the Data Grid is from terabytes to petabytes. Effective scheduling of jobs is necessary in such a system to use available resources such as computational, storage and network efficiently. If a job is assigned to a site where all the required data are available, then it would have no data transmission delay or reduced turnaround time. Of course, a scheduler should also consider parameters such as CPU workload, features of computational capability, network load, resource utilization and response time. Replication is a practical and efficient method to manage large data by storing data in a wise manner. Generally, replication algorithms are either static or dynamic. In static approaches the created replica will exist in the same place till the user deletes it manually or its duration is expired. On the other hand, dynamic strategies create and delete replicas according to the changes in grid environments, i.e. users' file access patterns. Storing replicas close to the users or grid computation nodes improve response time, fault tolerance and decreases bandwidth consumption. Therefore, an appropriate replica management framework is critical in complex systems such as Data Grids that access large amounts of data.

Grid can manage the same level of computing power as a supercomputer does, but at a much reduced cost. Grid is related to a virtual supercomputer. However, we need to deal with about many conditions such as network status and resource status because the members of grid are connected by networks. Scheduling independent tasks on grid is more complicated. In order to utilize the power of grid computing completely, we need an efficient job scheduling algorithm to assign jobs to resources. This paper focuses on the efficient job scheduling considering the completion time of jobs and achieve better performance by minimizing the data access time and avoiding unnecessary replication in a grid environment.

Issues related to data replication

Data replication strategies have been widely used in Data Grids to replicate frequently accessed data to suitable sites. There are certain problems which a data replication strategy must address during replication:

• The dynamic behavior of a Grid. The nature of the Grid is very changeable and users can enter and quit a Grid at any time. The data replication strategy must be adaptive to the information of the grid environment in order to provide better results.

• Grid architecture. The replication algorithm is dependent on Grid architecture. It can be a multi-tier architecture, a tree like structure, a graph like topology, a peer to peer topology or a

hybrid model. The replication strategy is proposed according to the architecture of the Grid.

• Three important questions. Any replication algorithm has to answer some important questions such as, (1) which files should be replicated; (2) when and how many replicas should be created; and (3) where the replicas should be placed in the system [1].

• Available storage space. The size of data that are requested in a Data Grid is from terabytes to petabytes. Restricted by the storage capacity, it is essential to present an effective replica replacement.

• Cost of replication. Unfortunately, there is a price to be paid when data are replicated. The files of Grid environments that can be changed by Grid users might bring an important problem of maintaining data consistency among the various replicas distributed in different machines. Exactly when and how those changes need to be carried out specifies the price of replication. The replication strategy must check that the benefit of the replication is more than the cost of replication.

2. RELATED WORK

Dynamic FPLTF Scheduling Algorithm (DFPLTF) [2] is based on FPLTF scheduling algorithm and is modified to make the FPLTF scheduling algorithm more adaptive for grid environment. Most Fit Task First Scheduling Algorithm (MFTF) [3] mainly attempts to assign the most suitable resource to the task by a value called fitness. Ant Colony Optimization (ACO) [4] was used for solving the scheduling problem in grids in recent years. Xu et al. proposed a simple grid simulation architecture and modified the basic ant algorithm for job scheduling in grid [5]. The scheduling algorithm they proposed needs some information such as the number of CPUs, Million Instructions Per Second (MIPS) of every CPU for job scheduling. A resource must submit the information mentioned above to the resource monitor.

Park et al. [6] presented a Bandwidth Hierarchy based Replication (BHR) which decreases the data access time by maximizing network-level locality and avoiding network congestion. They split the sites into various regions, where network bandwidth between the regions is lower than the bandwidth within the regions. So if the required file is placed in the same region, its fetching time will be less. The BHR strategy has two deficiencies, first it eliminates if a replica exists within the region and, second, replicated files are located in all the requested sites not the appropriate sites. The BHR strategy has good performance only when the capacity of the storage element is small.

Modified BHR [7] is an extension of the BHR [6] strategy which replicates a file that has been accessed most and it may also be used in the near future. The modified BHR strategy has a main weakness; it will search all sites to find the best one for storing the replica. EHR places the replica in the best site of the requested region.

In [8] a dynamic replication strategy is proposed that is a modified version of the Fast Spread replication strategy [9] that holds valuable replicas while the other less important replicas are replaced with more important replicas. A dynamic threshold is used to determine if the requested replica should be stored at each node along its path to the requester. They claimed their algorithm has better performance comparing with Fast Spread with LRU and Fast Spread with LFU (Least Frequently Used).

In [10] the authors presented a data replication strategy that has a provable theoretical performance guarantee and can be implemented in a distributed and practical manner. They also proposed a distributed caching strategy, which can be easily adopted in a distributed system such as Data Grids. The key point of their distributed strategy is that when there are several replicas, each Grid site keeps track of its closest replica site. This can dramatically enhance Data Grid performance because transferring large-sized files is time and bandwidth consuming [11]. The results of the simulation demonstrated that the distributed replication algorithm significantly outperforms a popular existing replication strategy under various network parameters.

Andronikou et al. [12] proposed a set of interoperable new data replication strategies that take into account the infrastructural constraints as well as the 'importance' of the data. The presented system is scalable and the strategies can be easily implemented on a Grid environment to provide fast execution. The proposed QoS-aware dynamic replication strategy determines the number of replicas required based on data request, content importance and requested QoS. It also places the new replicas within the Grid environment according to the network bandwidth and the overhead that the replication technique presents. It can handle the dynamicity of the Grid system by increasing or decreasing the set of data replicas based on the number and the geography of the data demands.

Saadat et al. [13] presented a new dynamic data replication strategy which is called Pre-fetching based Dynamic Data Replication Algorithm in Data Grids (PDDRA). PDDRA predicts future requirements of Grid sites and pre-replicates them before the needs are requested. This prediction is done based on the past file access history of the Grid sites. So when a Grid site requests a set of files, it will get them locally. The simulation results show that this strategy improves in terms of job execution time, effective network usage, number of replications, hit ratio and percentage of storage filled. But the problem of selecting best replicas when various sites hold replicas of datasets for the users' running jobs, i.e. best replica selection, has not been studied in this paper.

Taheri et al. [14] proposed a new Bee Colony based optimization strategy, called Job Data Scheduling using Bee Colony (JDS-BC). JDS-BC has two collaborating operations to efficiently schedule jobs onto computational elements and replicate data sets on storage elements in a system so that the two independent, and in many cases conflicting, objectives (i.e., makespan and transfer time of all data files) of such heterogeneous systems are concurrently decreased. Three tailor-made test Grids varying from small to large are applied to evaluate the performance of JDS-BC and compare it with other strategies. Results showed JDS-BC's superiority under differesnt operating scenarios.JDS-BC also presented a balanced decision making behavior, where it occasionally relaxes one of its objectives (e.g., transfer time) to obtain more from optimizing the other one (e.g., makespan).

3. PROPOSED WORK

The proposed algorithm has been described in three parts:

Replica selection: After a job is scheduled to site j, the requested data will be transferred to site j to become replicas. When different sites have replicas of a file, there is a significant benefit realized by selecting the best replica. The response time is an essential parameter that influences the replica selection and thus the job turnaround time. Each storage medium has many requests at the same time and the storage can respond only to one request at a time. Thus, there are requests waiting in the queue. The EHR algorithm selects the best replica location in a minimum response time that can be estimated by considering the data transfer time and the number of requests that are waiting in the storage queue. So, for replica selection: if the file is duplicated in the same LAN, then it will create a list of candidate replicas and selects the site that has the least number of requests. If the file does not exist in

the same LAN, then EHR searches within the local region. If the file is duplicated in the same region, then it will create a list of candidate replicas and selects a replica with the least number of requests. Otherwise it generates a list of replicas that are available in other regions, and from this list it selects the replica that has the least number of requests.

Replica placement: The EHR algorithm places the replica in the Best Storage Element (BSE). To select the BSE, EHR finds SE with minimum Value-SE (VSE) in the requested region. In the calculation of VSE the frequency of requests of the replica and the last time the replica was requested are considered. These parameters are important because they give an indication of the probability of requesting the replica again:

$$\mathrm{VSE} = (\mathrm{CT} - \mathrm{LT_i}) + \frac{1}{\mathrm{FR_i}}$$

where CT is the current time, LT_i is the last request time of replica i, and FR_i the frequency of requests of the replica i. The unit of time in CT and LT is the minute.

Replica management: If enough storage space exists in the BSE, the selected file is replicated. Otherwise if the file is available in the local LAN, then the file is accessed remotely. Now, if enough space for replication does not exist and the requested file is not available in the same LAN, one or more files should be deleted using the following rules:

• Generate a LRU sorted list of replicas that are both available in the site as well as the local LAN. Now start deleting files from the above list till space is available for the replica.

• If space is still insufficient, then repeat the previous step for each LAN in the current region randomly. In other words, generate a LRU sorted list of replicas that are both available in the site as well as the local region.

• If space is still insufficient, a group of replicas (that contain one or more replicas) need to be deleted. In this step using LRU may delete some valuable files that may not be available in the local region and may be needed in the future. Therefore, such deletions will result in a high cost of transfer.

In ASJS [15], users can submit different types of jobs at the same time containing computing-intensive jobs or data-intensive jobs. The computing-intensive job means that jobs need lots of computing power to complete and the data-intensive job means that the resource needs to take lots of bandwidth to transmit files. A user gives the specification of jobs such as types of jobs, number of computing-intensive jobs and number of data-intensive jobs on the interface provided by the Portal in the beginning. When the user completes the job description, the Portal will send the request to the Job Scheduler and Job Scheduler will determine the types of jobs and decide the weight value of cluster score function in initialization. After initializing the cluster score of each cluster, the Job Scheduler will select the resource with the best computing power in the cluster with the highest cluster score and assign the job to the resource. After Job Scheduler sends the job to the resource selected, it continues to schedule the next job until completing all jobs. After a resource receives a job, it starts to execute. The status of the resource will change, and therefore local update will be applied to adjust the cluster score of the cluster containing the resource. Once a job is completed by a resource, the result will be sent back and stored in the Information Server. At the same time, the global update will be applied to adjust the status of the entire grid system. Fig 1. Shows the overall architecture diagram.

After identifying the types of jobs, Job Scheduler will initialize the cluster score of each cluster. The cluster score is defined below:

$$\mathrm{CS_i} = \alpha \times \mathrm{ATP_i} + \beta \times \mathrm{ACP_i}$$

Where CS_i is the cluster score for cluster i, α and β are the weight value of ATP_i and ACP_i respectively, the sum of a and b is 1, ATP_i and ACP_i are the average transmission power and average computing power of cluster i respectively. ATP_i means the average available bandwidth the cluster i can supply to the job and is defined as:

$$\mathrm{ATP_i} = \frac{\sum_{j=1}^{m} \mathrm{Bandwidth\_available_{i,j}}}{m-1}, i \neq j$$

Where $\mathrm{Bandwidth\_available_{i,j}}$ is the available bandwidth between cluster i and cluster j, m is the number of clusters in the entire grid system. Similarly, ACP_i means the average available CPU power cluster i can supply to the job and is defined as:

$$\mathrm{ACP_i} = \frac{\sum_{k=1}^{n} \mathrm{CPU_{Speed_k}} \times (1 - \mathrm{load_k})}{n}$$

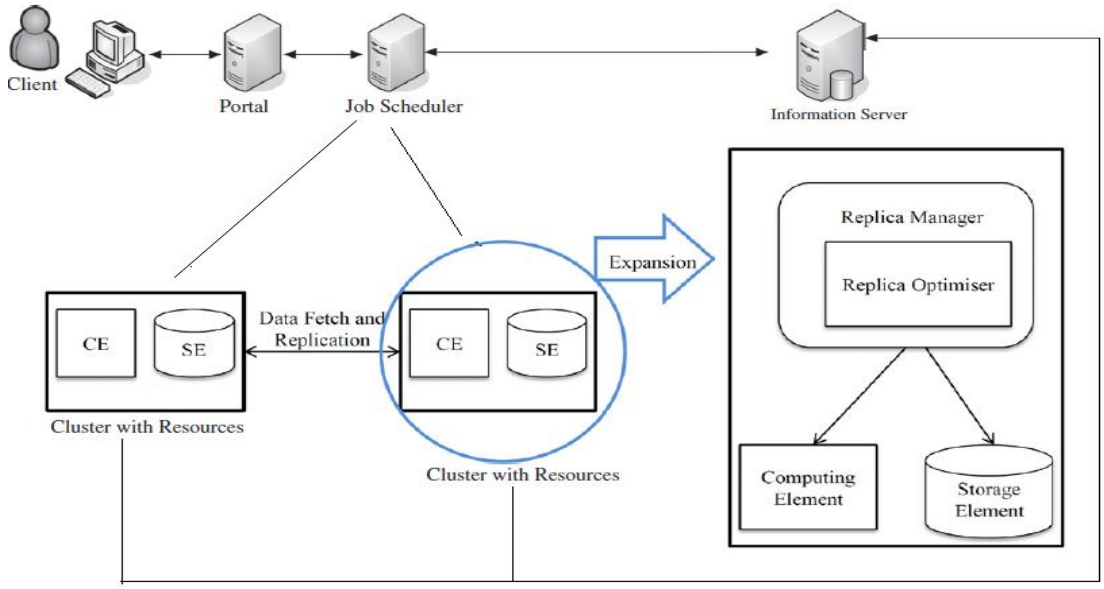

Fig 1. Architecture

Where CPU_speed_k is the CPU speeds of resource k in cluster i, $load_k$ is the current load of the resource k in cluster i, n is the number of resources in cluster i. Also let

$$CP_k = CPU\_Speed_k \times (1 - load_k)$$

CP_k indicates the available computing power of resource k.

Because the transmission power and the computing power of a resource will actually affect the performance of job execution, we use these two factors for job scheduling. Since the bandwidth between resources in the same cluster is usually very large, we only consider the bandwidth between different clusters. Another point is that the scale of the value of ATP_i and ACP_i are different.

Local update and global update are used to adjust the score. After a job is submitted to a resource, the status of the resource will change and local update will be applied to adjust the cluster score of the cluster containing the resource. What local update does is to get the available CPU percentage from Information Server and recalculate the ACP, ATP and CS of the cluster. After a job is completed by a resource, global update will get information of all resources in the entire grid system and recalculate the ACP, ATP and CS of all clusters.

4. EXPERIMENTAL RESULTS

With OptorSim, it is possible to simulate any Grid topology and replication strategy. So OptorSim code has been modified to implement the hierarchical structure, since it uses a flat network structure. It is assumed the network has three regions and on average two LANs in every region. The storage capacity of the master site is 200 GB, and the storage capacity of all other sites is 40 GB. We compare results using the parameters like mean job time based on varying size of files, Mean job time by varying the storage size and Mean job time based on varying number of jobs.

Fig 2. Mean job time based on varying number of jobs

Fig. 2 displays the mean job time based on the changing number of jobs for eight algorithms. The mean job execution time of EHR is lower by 38% compared to the ASJS algorithm for 300 jobs. According to the temporal and geographical locality, EHR places the replica in the best site. Also, it can minimize access latency by selecting the best replica. It is clear that as the job number increases, EHR is able to process the jobs in the lowest mean time in comparison with other methods. It is similar to a real Grid environment where a lot of jobs should be executed.

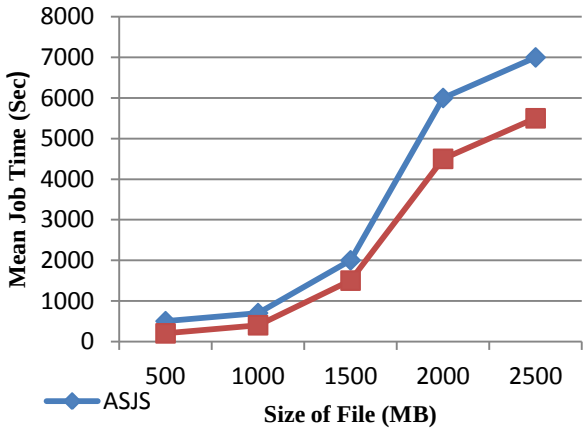

Fig 3. Mean job time based on varying size of files

Fig. 3 shows the effect of file size on the execution time for 300 jobs. As file size increases, storage media require more time for servicing. Here EHR outperforms the other methods as file size increases. The main reason is that the replica is placed in the best site by considering the frequency of requests of the replica and the last time the replica was requested. So the performance of the proposed algorithm is improved by minimizing the data access time and avoiding unnecessary replication. It is necessary to note that in data-intensive applications, the locations of data required by the job affects the performance significantly.

When the Grid storage size is fixed, the number of replicas in the Grid will be limited. Fig. 4 illustrates the effect of the size of a storage element on the mean job time for eight algorithms. EHR takes the least mean job execution time among strategies. As the size of storage space decrease in Grid sites, EHR greatly outperforms other strategies. If the available storage for replication is not enough, the proposed algorithm will only replicate those files that are not available in the nearby sites. It does not delete valuable files that may not be available in the local region and may be needed in the future. But if the available storage for replication is enough, all algorithms have the same performance.

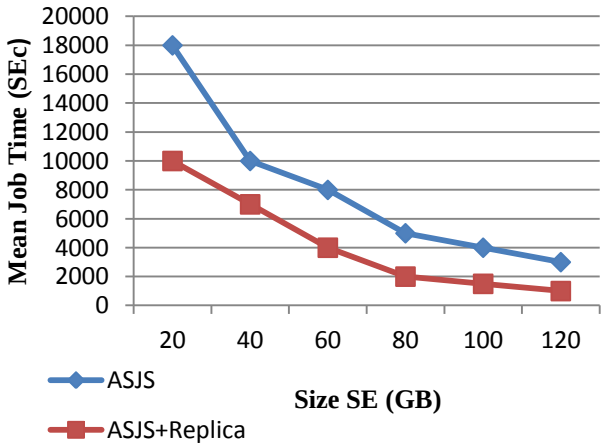

Fig 4. Mean job time by varying the storage size

5. CONCLUSION AND FUTURE WORK

A dynamic data replication strategy, called Effective Hierarchical Replication (EHR) is proposed. EHR selects the best site for placing the replica by considering the frequency of requests and the last time the replica was requested. It also selects the best replica location for execution of jobs by considering the number of requests that are waiting in the storage queue, and data transfer time. If enough space for replication does not exist EHR deletes files in two steps when free space is not enough for the new replica. First, it deletes those files with minimum time for transfer (i.e. only files that exist in the local LAN and local region). Second, if space is still insufficient then it uses an Economic Model.

We propose a replica strategy with adaptive scoring method to schedule jobs in grid environment. ASJS selects the fittest resource to execute a job according to the status of resources. Local and global update rules are applied to get the newest status of each resource. Local update rule updates the status of the resource and cluster which are selected to execute the job after assigning the job and the Job Scheduler uses the newest information to assign the next job. Global update rule updates the status of each resource and cluster after a job is completed by a resource. It supplies the Job Scheduler the newest information of all resources and clusters such that the Job Scheduler can select the fittest resource for the next job.

To evaluate the efficiency of the proposed strategies, we used the Grid simulator OptorSim that is configured. The experimental results show that ASJS+Replica is capable of decreasing completion time of jobs and the performance of ASJS+Replica is better than other methods.

In the future, we plan to test our simulation results on real Data Grids. We will also try to investigate dynamic replica maintenance issues such as replica consistency. In the longer term, we would like to consider the set of QoS factors taken into account for dynamic replication, including both service provider and client related requirements.

REFERENCES

[1] K. Sashi, A. Thanamani, Dynamic replication in a Data Grid using a modified BHR region based algorithm, Future Generation Computer Systems 27 (2) (2011) 202–210.

[2] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system, Journal of Parallel and Distributed Computing 59 (1999) 107–131.

[3] Sheng-De Wang, I-Tar Hsu, Zheng-Yi Huang, Dynamic scheduling methods for computational grid environment, International Conference on Parallel and Distributed Systems 1 (2005) 22–28.

[4] E. Salari, K. Eshghi, An ACO algorithm for graph coloring problem, in: Congress on Computational Intelligence Methods and Applications, December 2005, pp. 15–17.

[5] Hui Yuan, Xue Qin, Ximg Li, Ming-Hui Wu, An improved ant algorithm for job scheduling in gird computing, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 5, 18–21 August, 2005, pp. 2957-2967.

[6] S.-M. Park, J.-H. Kim, Y.-B. Go, W.-S. Yoon, Dynamic grid replication strategy based on internet hierarchy, in: International Workshop on Grid and Cooperative Computing, in: Lecture Note in Computer Science, vol. 1001, 2003,pp. 1324–1331.

[7] K. Sashi, A.S. Thanamani, Dynamic replication in a Data Grid using a Modified BHR region based algorithm, Future Generation Computer Systems 27 (2011) 202–210.

[8] M. Bsoul, A. Khasawneh, E.E. Abdallah, Y. Kilani, Enhanced fast spread replication strategy for Data Grid, Journal of Network and Computer Applications 34 (2011) 575–580.

[9] K. Ranganathana, I. Foster, Identifying dynamic replication strategies for a high performance Data Grid, in: Proceedings of the International Grid Computing Workshop, 2001, pp. 75–86.

[10] D.T. Nukarapu, B. Tang, L. Wang, S. Lu, Data replication in data intensive scientific applications with performance guarantee, IEEE Transactions on Parallel and Distributed Systems 22 (2011).

[11] A. Chervenak, R. Schuler, M. Ripeanu, M.A. Amer, S. Bharathi, I. Foster, C. Kesselman, The globus replica location service: design and experience, IEEE Transactions on Parallel and Distributed Systems 20 (2009) 1260–1272.

[12] V. Andronikou, K. Mamouras, K. Tserpes, D. Kyriazis, T. Varvarigou, Dynamic QoS-aware data replication in grid environments based on data importance, Future Generation Computer Systems 28 (3) (2012) 544–553.

[13] N. Saadat, A.M. Rahmani, PDDRA: a new pre-fetching based dynamic data replication algorithm in Data Grids, Future Generation Computer Systems 28(7) (2011) 1045–1057.

[14] J. Taheri, Y.C. Lee, A.Y. Zomaya, H.J. Siegel, A bee colony based optimization approach for simultaneous job scheduling and data replication in grid environments, Computers & Operations Research (2011) http://dx.doi.org/10.1016/j.cor.2011.11.012.

[15] Ruay-Shiung Chang, Chih-Yuan Lin, Chun-Fu Lin, "An Adaptive Scoring Job Scheduling algorithm for grid computing", Future Generation Computer Systems 207 (2012) 79–89.