A New Job Scheduling in Data Grid Environment Based on Data and Computational Resource Availability

Najme Mansouri[1]

1-Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran

ABSTRACT

Data Grid is an infrastructure that controls huge amount of data files, and provides intensive computational resources across geographically distributed collaboration. The heterogeneity and geographic dispersion of grid resources and applications place some complex problems such as job scheduling. Most existing scheduling algorithms in Grids only focus on one kind of Grid jobs which can be data-intensive or computation-intensive. However, only considering one kind of jobs in scheduling does not result in suitable scheduling in the viewpoint of all systems, and sometimes causes wasting of resources on the other side. To address the challenge of simultaneously considering both kinds of jobs, a new Integrated Job Scheduling Strategy (IJSS) is proposed in this paper. At one hand, the IJSS algorithm considers both data and computational resource availability of the network, and on the other hand, considering the corresponding requirements of each job, it determines a value called W to the job. Using the W value, the importance of two aspects (being data or computation intensive) for each job is determined, and then the job is assigned to the available resources. The simulation results with OptorSim show that IJSS outperforms comparing to the existing algorithms mentioned in literature as number of jobs increases.

KEYWORDS

Data Grid, Scheduling, Access Pattern, Simulation

1. INTRODUCTION

Nowadays, in different scientific disciplines, large data sets are becoming an important part of shared resources. Most applications especially in scientific and engineering fields tend to be data-intensive and/or computation-intensive. Due to the fact that it is impossible to manage these applications in a central server, Grid technology has been proposed as a suitable infrastructure to replace it. All jobs in such environment will compete for some resources and this is possible to distribute the load disproportionately among the Grid sites. One of the most important challenges in Grid is job scheduling problem. Indeed, determining the optimal schedule for a Grid environment which can distribute the sent jobs to the Grid resources to optimize a specify measure is a well-known NP-complete problem [1]. To overcome this difficulty, many heuristic strategies have been presented to appropriately schedule jobs among resources [2-7]. None of these types of scheduling strategies can be clearly claimed to propose optimal solutions. Moreover, the current scheduling strategies [8-16] are immutable to changing schedules and behave like static time-dependent Grid systems. These schedulers cannot consider the input parameters such as network features and data location at runtime. The job scheduler should take into consideration input constraints such as data location, data size, site availability, network features, computation power and various optimization criteria in making scheduling decisions.

1.1 MOTIVATION

Generally speaking, the applications can be divided into two basic classes, data-intensive and computation-intensive applications. Data-intensive applications dedicate most of their operation time to access data (like High Energy Physics, Genetic, and Earth Observation), however, computation-intensive applications devote most of their operation time to compute and process on data (like Image processing). The Grid scheduling decisions are often made on the basis of jobs being either data or computation intensive: in data intensive states jobs may be pushed to the data and in computation intensive states data may be pulled to the jobs. This type of scheduling, in which there is no consideration of network features, can lead to performance reduction in a Grid environment and may result in large processing queues and job execution delays due to site overloads. Furthermore, previous strategies have been based on so-called greedy algorithms where a job is assigned to a 'best' resource without evaluating the global cost of this action. However, this can lead to a skewing in the allocation of resources and can

result in large queues, reduced performance and throughput degradation for the other jobs.

Also some application is both data-intensive and computation-intensive (like Galaxy Formation and Evolution). For example, most SAMR applications are computation intensive; belong to high performance scientific computing capacity. Due to deep levels of adaptive refinements, SAMR applications can also be communication intensive. In some cases, when dealing with large amounts of data, SAMR applications can fall into data intensive category. Experiments show that during the entire course of execution, SAMR applications may run in different execution modes as the simulated physical phenomena evolve.

The nature of applications can also affect the result of the scheduling and should be used during scheduling decision. Generally speaking, the applications can be classified into two common classes, data-intensive and computation-intensive applications. Data-intensive applications devote most of their operation time to access data [17-19], however, computation-intensive applications dedicate most of their operation time to process on data [20]. In fact, almost no application belongs to one of these two categories specifically; nevertheless it requires data/computational resources proportionally to be run. In other words, most applications are both data-intensive and computation-intensive. However, the proportion between being data and computation intensive differs among applications. Focusing on only one of these aspects causes important problems, since the other one is not negligible. At one hand, evaluating only data-intensive aspect causes a waste of computational power; on the other hand, evaluating only computation-intensive aspect leads to a waste of network resources such as bandwidth. We propose a new Integrated Job Scheduling Strategy (IJSS) that addresses these problems. The IJSS algorithm is a way to simultaneously use data-intensive and computation-intensive dimensions of the job, while taking into account the same characteristics of the available Grid environment. The scheduler can make good selections by considering the changing state of the network, the locality and the size of data and computational power. In other words, the scheduler needs to schedule any sent job adaptively based on the present state of the network as well as the job. The simulation results show that considerable performance improvements can be gained by adopting the IJSS scheduling approach.

The rest of the paper is organized as follows. Section 2 explains phases of Grid job scheduling. Section 3 introduces related work of this study. Section 4 presents

the proposed job scheduling algorithms. Section 5 explains elements of grid simulation. We show and analyze the simulation results in section 6. Finally, section 7 concludes the paper and suggests some directions for future work.

2. PHASES OF GRID JOB SCHEDULING

The Grid scheduling process can be divided into three main stages [21]: resource discovery where a list of possible resources is generated, resource selection where a set of resources is selected and task execution where the tasks are run and monitored. Figure 1 presents the three main steps and the phases that make them.

Resource Discovery: the first phase is to determine which resources are available. The first step consists in specifying the set of resources that a user has access to. At the end of this phase, the user will have a set of resources that he/she can access. The next stage is the application requirement definition. In this step the user determines a list of requirements for the job in order to filter the set of resources. The next stage is to do a minimal requirement filtering. The goal is to remove the resources that do not meet the minimal requirements.

System Selection: in this phase the goal is to select a single resource to schedule the job. This is done in two steps: gather dynamic information and system selection. Gather dynamic information is important in order to make the best mapping between job and resource. The system selection consists in choosing a resource with the gathered information.

Task Execution: the first stage, advanced reservation, is optional. The aim is to make the best use of the system. Advanced reservation difficulty related to the considered resource. When the resource or resources are selected, the task needs to be submitted. The next stage is preparation. In this step a list of operations takes place to prepare the resource to execute the task. The following stage is monitoring. Once the task is started it is essential to keep track of its progress. By monitoring tasks, the scheduler can decide that a given task in not making progress and may reschedule it. The next stage is job completion were the user is informed when a task or job completed. The last stage is the cleanup were temporary files are deleted and the user gets information from the resource that will be used to analyze the results finally

Complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

3. RELATED WORK

Generally, job scheduling in Grid has been studied from the perspective of computational Grid [22-23]. In Data Grid, effective scheduling policy should consider both computational and data storage resources.
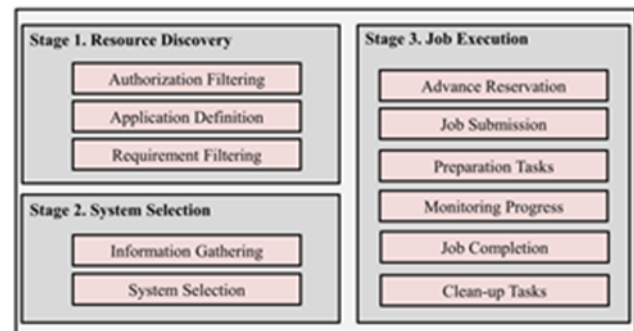

Fig. 1. Grid scheduling phases.

Foster et al. [24-25] proposed six distinct replica strategies for a multi-tier data: No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread. They also introduced three types of localities, namely:

- Temporal locality: The files accessed recently are much possible to be requested again shortly.

- Geographical locality: The files accessed recently by a client are probably to be requested by adjacent clients, too

- Spatial locality: The related files to recently accessed file are likely to be requested in near future.

They evaluated these strategies with different data patterns: access pattern with no locality, data access with a small degree of temporal locality and finally data access with a small degree of temporal and geographical locality. The results of simulations indicate that different access pattern needs different replica strategies. Cascading and Fast Spread performed the best in the simulations. They have presented in another work [24] the problem of scheduling job and data movement operations in a distributed "Data Grid" environment to identify both general principles and specific strategy that can be used to improve system utilization and/or response times. They have also proposed framework with four different job scheduling algorithms, as follows:

(1) JobRadom: select a site randomly, (2) JobLeastLoaded: select a site where has the least number of jobs waiting to run, (3) JobDataPresent: select a site where has requested data, and (4) JobLocally: run jobs locally. These job scheduling strategies are combined with

three various replication algorithms: (1) DataDoNothing: there is no replication and data.

May be fetched from a remote site for a particular job, (2) DataRandom: when popularity of the file exceeds a threshold, a replica is created at a random site, (3) DataLeastLoad: when the threshold for a file exceeds, a replica is placed at the least loaded site. They can enhance performance by scheduling jobs where data is located and using a replication policy that periodically creates new replicas of popular datasets at each site. The results also show that while it is important to consider the impact of replication on the scheduling strategy, it is not always necessary to couple data movement and computation scheduling.

Chang et al. [26] developed the Hierarchical Cluster Scheduling algorithm (HCS) and the Hierarchical Replication Strategy (HRS) to enhance the data access efficiencies in a Grid. HCS considers the locations of required data, the access cost and the job queue length of a computing node. It also takes into account hierarchical cluster Grid structure and all of data replicas owned by a cluster. The HRS replication algorithm uses the concept of "network locality" as a Bandwidth Hierarchy based Replication (BHR) strategy. HCS scheduling along with HRS replica strategy improves data access time and the amount of inter-cluster communications in comparison with others scheduling algorithms and replication strategies.

A replication algorithm for a 3-level hierarchy structure and a scheduling algorithm are proposed. Horri et al. [27] considered a hierarchical network structure that has three levels. In their proposed replication method among the candidate replicas they select the one that has the highest bandwidth to the requested file. Similarly, it uses the same technique for file deletion. This leads to a better performance comparing with LRU (Least Recently Used) method. For efficient scheduling, 3-level scheduling (3LS) algorithm selects the best region, LAN and site respectively. Best region (LAN, site) is a region (LAN, site) with most of the requested files. This will significantly reduce total transfer time, and consequently the network traffic.

Mansouri et al. [28] proposed a new job scheduling algorithm, called Combine Scheduling Strategy (CSS). CSS first selects the appropriate region, next selects the appropriate LAN in that region (i.e. available maximum requested files) and finally selects the appropriate site in that LAN by considering number of jobs waiting in the queue, location of required data and the computing capacity of sites. Simulation results show that CSS takes less job execution time than other strategies especially when number of jobs or size of the files or both increases.

Zhang et al. [29] investigated the impact of parallel download on job scheduling performance in Data Grid. In their proposed method, data file is divided into fragments and placed on various replica servers. The replica fragments are transferred concurrently from several replica servers when a replica is requested. They have evaluated the performance of the parallel downloading system with the traditional non-parallel downloading systems using different Grid scheduling strategies. The results of simulation indicated that it can improve the network bandwidth consumption and speed up the data access. When network bandwidth is relatively low and computing power is relatively high, parallel downloading has been effective.

Mohamed et al. [30] proposed the Close-to-Files (CF) scheduling algorithm that schedules a job on sites with enough processing capacity close to a site where data are present. CF uses an exhaustive algorithm to check all combinations of computing sites and data sites to select a site with the minimum computation and transmission delay. CF has a better performance in comparison with WF (Worst-Fit) job placement strategy, which allocates jobs on the sites with the largest number of idle processors.

Tang et al. [31] proposed a framework that supports job scheduling and data replication in Data Grid. Several replication strategies and their performance are evaluated with some scheduling heuristics. The computing sites are organized into individual domains according to the network connection, and a replica server is placed in each domain. The data replication algorithms increase the scheduling performance from the perspective of job computing time. In particular, the combination of shortest turnaround time scheduling heuristic (STT) and centralized dynamic replication indicates prominent performance.

In [32] the problem of co-scheduling job dispatching and data replication in large distributed systems in an integrated manner is presented. They used a massively-parallel computation model that contains a collection of heterogeneous independent jobs with no inter job communication. The proposed model has three variables within a job scheduling system: the job order in the scheduler queue, the assignment of jobs to the nodes, and the assignment of data replicas to data stores. Finding the optimal tuple requires an exhaustive search and it is costly because the solution space is very big. The results show that deploying a genetic search algorithm has the potential

to achieve a better performance than traditional allocation methods.

Kumar et al. [33] showed why network characteristics, data locations of input files, and disk read speed of data sources must be taken into account when scheduling data intensive jobs, not only to minimize file staging (data transfer) time over network, but also to reduce turnaround and waiting time of jobs in Grid environment. They presented Network and Data Location Aware Scheduling (NDAS) algorithm. The presented algorithm is evaluated by improving the existing GridWay MetaScheduler with the new scheduling algorithm. The excremental results regarding the influence of the network characteristics, data locations, disk latency of data source, and jobs types variability are presented, showing that the enhanced GridWay can perform better job scheduling resulting to lower data transfer and turnaround time.

Wu et al. [34] presented an integrated security-aware job scheduling strategy for computational grids. They proposed a genetic algorithm for job scheduling to address the heterogeneity of fault-tolerance mechanisms problem in a computational grid. They considered that the system supports four types fault-tolerance methods, including the job retry, the job migration without check pointing, the job migration with check pointing, and the job replication mechanisms. Because each fault-tolerance method has different requirements for gene encoding, they also proposed a novel chromosome encoding approach to integrate the four kinds of mechanisms in a chromosome. Simulation results demonstrated that their algorithm has shorter makes span and more excellent efficiencies on improving the job failure rate than the Min–Min and suffer age algorithms. But the main limitation of the algorithm is that it considers only the fault-tolerance strategy. Further, they focus on the improvement of the system performance and they do not address the locality and the size of data.

Hemamalini et al. [35] presented an assessment of task scheduling algorithm based on various factors such as communication cost, and task duplications in a heterogeneous grid computing environment. Some task scheduling algorithms improves the make span but it causes a load imbalance. Many tasks scheduling algorithms only focus on one kind of Grid jobs which can be data-intensive or computation-intensive. Therefore, we present a way to simultaneously use of data-intensive and computation-intensive dimensions of the job, while taking into account the same characteristics of the available Grid environment.

Amalarethinam et al. [36] listed a sequence of classification and comparison of the scheduling policies in grid computing. They presented a package of reviews taking various factors that are having a greater influence while scheduling the jobs. In this survey, the algorithms are keyed with communication cost, execution time length, error factors, and task duplications. To the best of our knowledge, job scheduling problem has been intensively investigated in grid computing environment while, the job scheduling issue in data grid environment has not been thoroughly studied yet. Also the most available scheduling algorithms in grid computing only focus on computation-intensive job. We propose a new Integrated Job Scheduling Strategy (IJSS) that addresses these problems.

4. NETWORK AND DATA LOCATION AWARE SCHEDULING (NDAS) ALGORITHM

To select a best site, a parallel strategy is proposed as shown in Fig 2.

A. Transfer Time

Let Bji is the bandwidth from site Sj to the site that fi resides. PropagationDelayij is propagation delay / network latency (in seconds) from site Sj to site Si. Then transfer time for fi (TransferTimefi) is obtained by

$$TransferTime_{fi} = PropagationDelay_{ij} + \left(\left| fi \right| * 8 \right) / B_{ji} \qquad (1)$$

Let Jx = {f1, f2, fm} be the m required files for job x. Now estimated file staging (data transfer) time of job x when scheduled on site Sj (JobTimex,j) is given as :

$$JobTime_{x,j} = \sum_{i=1}^{m} Min(TransferTime_i) \qquad (2)$$

Replica selection is crucial to data intensive scheduling; it depends on the network characteristics and an optimized replica selection leads to an optimized data intensive scheduling.

These considerations not only improved the execution times of the jobs but also reduced the queue times of the jobs. So, if several sites have the replica of fi, it selects one that has a maximum Score.

$$Score = P^{BW} \times w_1 + P^{CPU} \times w_2 + P^{IO} \times w_3 \qquad (3)$$

where PBW represents the percentage of bandwidth available from the selected site to the site that the requested file resides, PCPU is the percentage CPU idle states of site that requested file resides, and PIO is the percentage of memory free space of site that requested file resides.
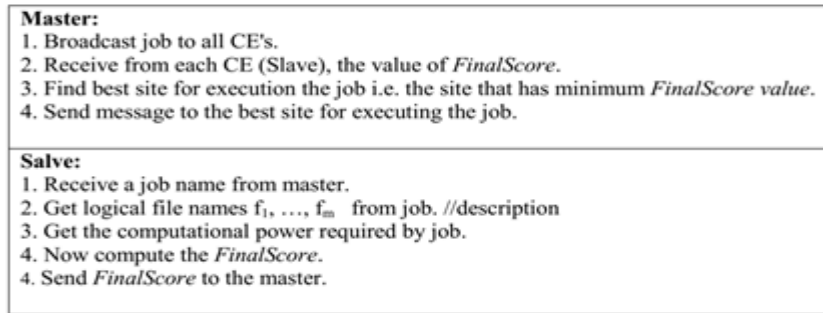
Master:
1. Broadcast job to all CE's.
2. Receive from each CE (Slave), the value of *FinalScore*.
3. Find best site for execution the job i.e. the site that has minimum *FinalScore value*.
4. Send message to the best site for executing the job.

Salve:
1. Receive a job name from master.
2. Get logical file names $f_1, \ldots, f_m$ from job. //description
3. Get the computational power required by job.
4. Now compute the *FinalScore*.
4. Send *FinalScore* to the master.

Fig. 2. A parallel execution flow of master and slave

$$w_1 + w_2 + w_3 = 1 \tag{4}$$

These weights can be set by the administrator of the Data Grid organization. According to different attributes of storage systems in data Grid node.

Let k is the number of jobs waiting in queue of site Sj. The value of TotalTimej for site Sj is calculated by

$$TotalTime_j = \sum_{x=1}^{k} JobTime_{x,j} \tag{5}$$

B. Computational Power

The processing power provided by resources (required for jobs) is described in the form of MIPS (MI). Therefore, the total time required for the job Jx to be completed in the resource Sj can be calculated by Eq. (6).

$$ComputingScore = \frac{CP_x}{CP_j} \tag{6}$$

where CPj is the computational power provided by the computational resource Cj and CPx is the computational power required by job Jx. The ComputingScore is used as a score for fitness of the resource Cj for job Jx. The available information about each job send to the environment is stored in two areas. The first one contains information about needed data files, so we can obtain the total size of data files, and the second one gives information about the total computational power needed by the job in terms of MI. The main goal at this stage is to calculate the proportion of being data-intensive to being computation-intensive, while considering the availability of resources in each area. Hence, the strategy needs to jointly consider both required and provided resources, and then estimate a value for scheduler to show how much the submitted job is generally data/computation intensive in the context of available grid environment.

To achieve this, the strategy first determines the expected value of the provided computational power using Eq. (7).

$$ComputationPower = \frac{\sum_{i=1}^{N} Cp_i}{N} \tag{7}$$

where, N is the number of sites. To find the corresponding value for data-intensive aspect of the submitted job, the strategy needs to apply an equivalent mean operation on network links. Eq. (5) obtains this value by averaging on time needed to collect a specific set of data files for each site.

$$TotalTransferTime = \frac{\sum_{i=1}^{N} TotalTime_i}{N} \tag{8}$$

C. Final Cost

Finally, the factor W is determined by using Eq. (9) and Eq. (10) for a given job is.

$$TT = \frac{TotalTime_i}{TotalTransferTime}$$
$$CC = \frac{CP_i}{ComputationPower} \tag{9}$$

$$W = \frac{CC}{CC + TT} \tag{10}$$

when the IJSS strategy is executed for a submitted job, both Total Time and ComputingScore are determined for each site. Combining these two scores by affecting the factor W gives the Final Cost for all sites Eq. (11).

$$FinalCost(J,S) = (1-w) \times TotalTime + \tag{11}$$
$$w \times ComputingScore$$

The IJSS strategy chooses the site with minimum Final Cost and assigns the job to it.

5. ELEMENTS OF GRID SIMULATION

We have implemented the proposed strategy using OptorSim, a simulator for Data Grids. OptorSim was presented by the European Data Grid (EDG) project [37]. It provides users with the Data Grids simulated architecture and programming interfaces to analysis and validate their strategies. In order to obtain a realistic simulated environment, there are a number of components which are included in OptorSim. These include Computing Elements (CEs), Storage Elements (SEs), Resource Broker (RB), Replica Manager (RM), and Replica Optimizer (RO). Each site consists of zero or more CEs and zero or more SEs (as shown in Fig 3.)

6. EXPERIMENTS

In this section, network configuration and the simulation results are described.

A. Configuration

The study of our scheduling algorithm is carried out using a model of the EU Data Grid Testbed [37] sites and their associated network geometry as shown in Fig. 4. Initially all jobs are placed on CERN (European Organization for Nuclear Research) storage element. CERN contains an original copy of some data sample files that cannot be removed. Since all files are available in Site 0, so any job sent to this site does not require any file transfer. Therefore, in our simulation we only consider all CE sites except site 0. Each file is set to 1 GB. To record file transfer time and path, we changed OptorSim code. A job will typically request a set of logical filename(s) for data access. The order in which the files are requested is specified by the access pattern. We considered five different access patterns: sequential (files are accessed in the order stated in the job configuration file), random (files are accessed using a flat random distribution), unitary random (file requests are one element away from previous file request but the direction is random), Gaussian random walk (files are accessed using a Gaussian distribution), and Random Zipf access (given by $P_i = K/i_s$, where P_i is the frequency of the ith ranked item, K is the popularity of the most frequently accessed data item and S determines the shape of the distribution).

B. Simulation Results And Discussion

Eight scheduling strategies are considered, as follows:

The Random scheduler that schedules a job randomly.

The Shortest Queue scheduler that selects computing element that has the least number of jobs waiting in the queue.

The Access Cost scheduler that assigns the job to computing element where the file has the lowest access cost (cost to get all unavailable requested data files needed for executing job).
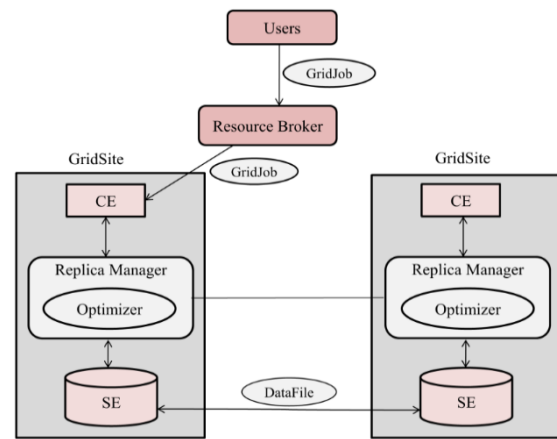

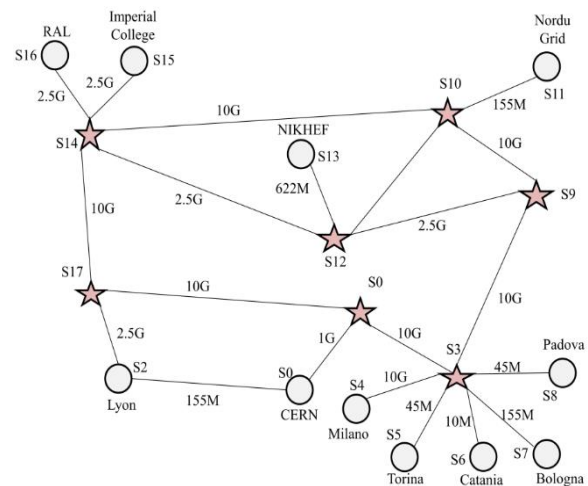**Fig. 3. OptorSim architecture.**

Fig. 4. The gird topology of EDG.

The Queue Access Cost scheduler that selects computing element with the smallest sum of the access cost for the job and the access costs for all of the jobs in the queue.

Hierarchical Cluster Scheduling (HCS) takes into account hierarchical cluster Grid structure and all of data replicas owned by a cluster. It schedules jobs to certain specific sites and specific cluster according to inter-cluster communication costs.

An OVERA3-level Scheduling (3LS) determines the most appropriate region, LAN and site respectively. An appropriate region (LAN, site) is a region that holds most of the requested files (from size point of view). I.e. most of the requested files are available in that region.

Network and Data location Aware Scheduling (NDAS) takes into account network characteristics, data locations of input files, and disk read speed of data sources in scheduling decision.

The Combine Scheduling Strategy (CSS) considers the number of jobs waiting in queue, the location of required data for the job and the computing capacity of sites.

Figure 5 depicts the Mean Job Time for different job scheduling algorithms with various access patterns. The mean job execution time is defined as the total time to run all the jobs divided by the number of jobs finished. The total time includes the time that elapses from when a job enters the queue in a site to await execution until the time when the job completes its processing and leaves the site. In Random scheduling the mean job execution time obviously increases because it doesn't consider any factors.

In Shortest Job Queue Scheduling each CE receives approximately the same number of jobs. If CE's have low network bandwidth, then file transfer time will be high and the overall job execution time will increase. Access Cost Scheduling selects a CE based on its access cost. CE's with lower access cost may receive large number of jobs to execute. So, overall performance is decreased. The Queue Access Cost considers not only the shortest job queue but also access cost. Therefore, the Queue Access Cost decreases total the job execution time. The mean job time is about 8% faster using HCS than using Queue Access Cost because HCS uses a hierarchical tree to schedule a job and minimizes the overhead of searching for the suitable site. The 3LS first selects the appropriate region (i.e. available maximum requested files), next selects the appropriate LAN in that region and finally selects the appropriate site in that LAN, therefore, job execution time decreases since it has minimum data transfer time. The mean job time is about 12% faster using CSS than using HCS because it schedules jobs close to the data whilst ensuring sites with a high network connectivity are not overloaded and sites with a poor connectivity are not left idle. It also takes into account hierarchical Grid structure and considers computational capability. The mean job time of IJSS is lower about 11% compared to the CSS algorithm. The reason is that it takes into account data, processing power and network characteristics when making scheduling decisions across different sites.

Figure 6 shows the queue time for nine scheduling strategies with different number of jobs. We changed the number of jobs for two important reasons: to monitor how the queue size increases over time and in which proportion

the scheduler submits the jobs (that is whether the jobs are sent to some particular site or to a number of CPUs at various locations depending on the queue size and the computing capability). It presents that queue time is almost proportional to execution time because if the job is executing and taking more time on the processor, the waiting time of the new job will also increase correspondingly since it will waste more time in the queue.

Although the execution time does not comprise queue times, a higher number of jobs executing at a site can influence the queue time. Moreover, increasing the number of jobs in the queue can affect the overall job completion times (i.e. the scheduling time, queuing time and execution time) of the new jobs. The queue time of the schedulers is very important in the Grid environment and it takes a large ratio of the job's overall time. Sometimes this is greater than the execution time if the resources are rare compared to the job frequency. In experimental setup of this work, we took only a single job queue and we considered that all jobs have the same priority.

Multi-queue and multi-priority job scenarios will be discussed later in future work. Figure 6 indicates that the queue grows with an increasing number of jobs and that the number of jobs waiting for the allocation of the processors for running also increases. From the figure it is clear that the IJSS scheduling strategy remarkably decreases the queue time of the jobs. The main reason is only those sites were selected for job placement which had fewest jobs in the queue and which were likely to quickly run the jobs once scheduled on that site were selected for job placement.
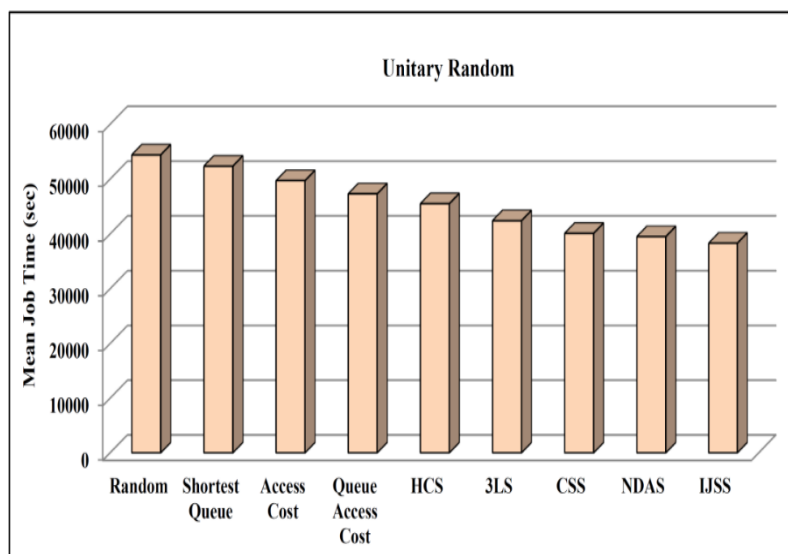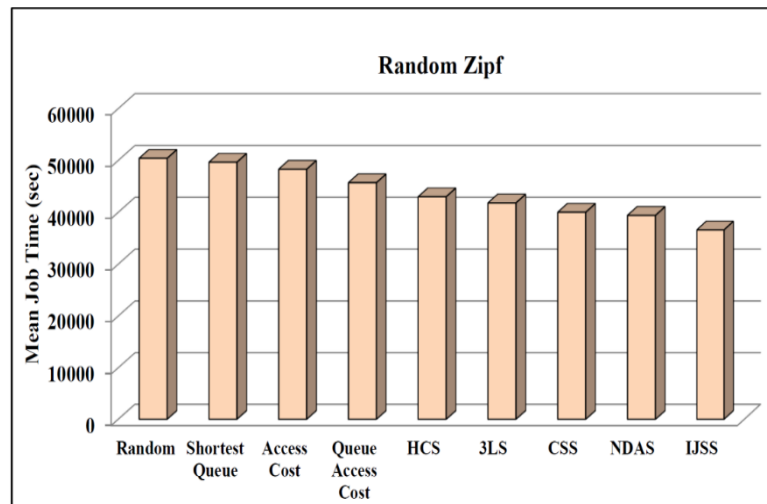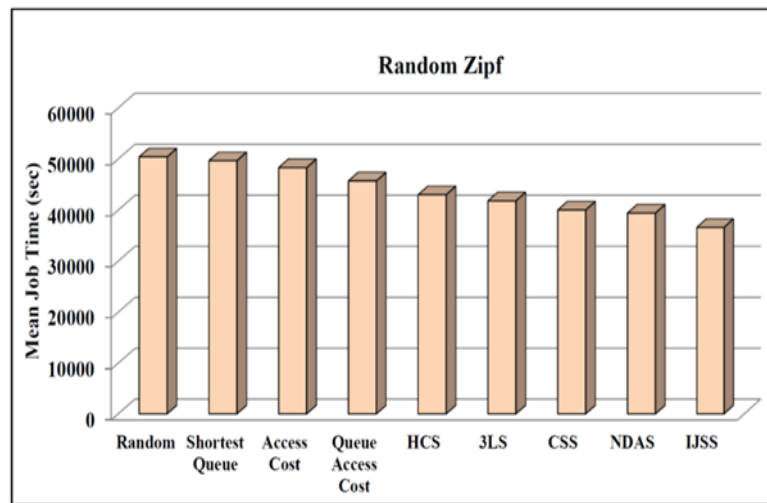
Figure 7 indicates execution times for various scheduling strategies. We see from the results obtained in Fig. 6 and 7 that both queue and execution times follow very similar trends. This is mainly due to the fact that IJSS preferentially chose those sites for job execution which could execute jobs fast.

The computing resource usage is shown in Fig. 8. It is the percentage of time that CEs are in active state. The IJSS has good computing resource usage because it completes all jobs first, so the CPUs are not idle most of the time. It can make intelligent decisions by considering the changing state of the network, and the pool of processing cycles.

The average number of intercommunications for a job execution is presented in Fig. 9. By choosing the best site with minimum cost, IJSS can decrease the cost of

intercommunications effectively as compared with the other job scheduling algorithms. Since IJSS schedules data-intensive jobs to certain specific sites according to

data transfer costs. Therefore, data-intensive jobs would be executed on a site with the most needed files.
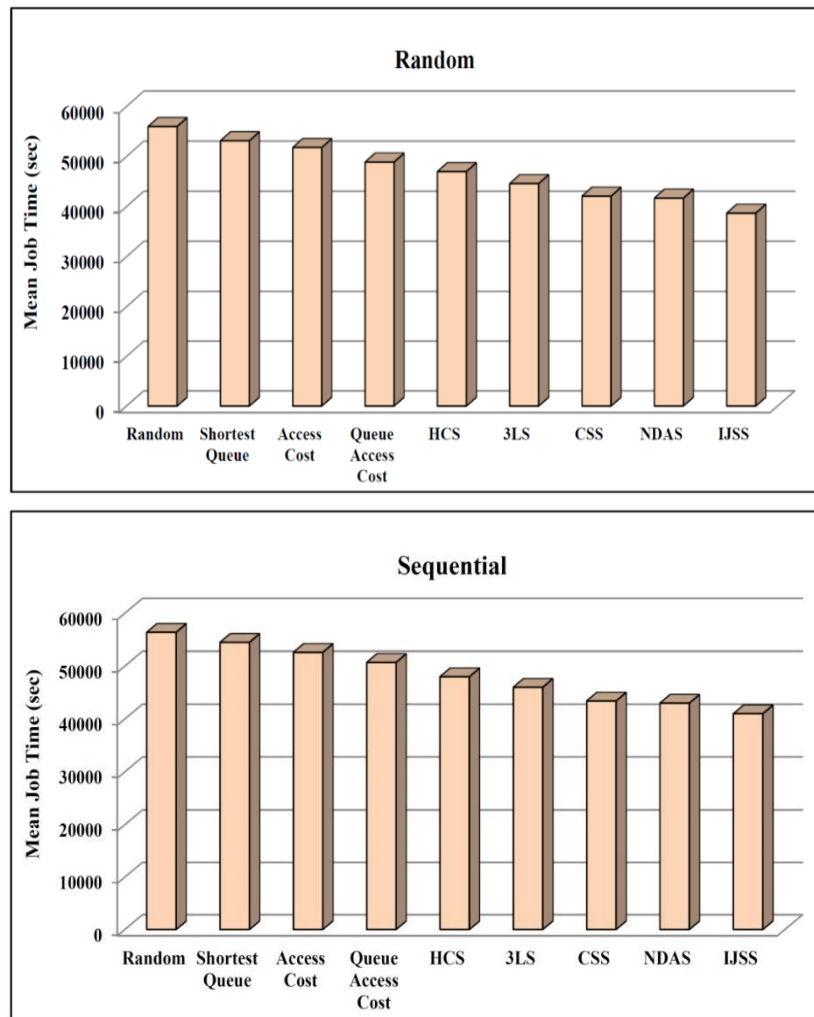
Amirkabir International Journal of Science & Research
(Modeling, Identification, Simulation & Control)
(AIJ-MISC)

Najme Mansouri

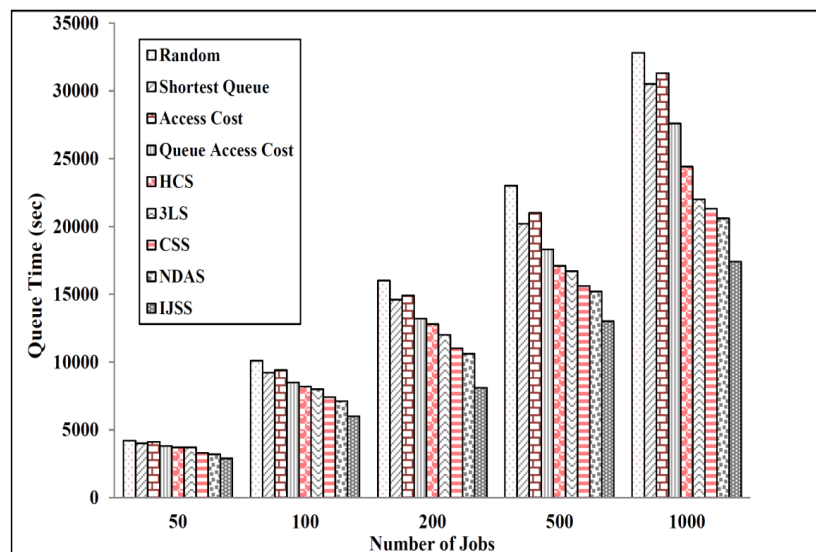**Fig. 5. Mean job Time for different access patterns.**

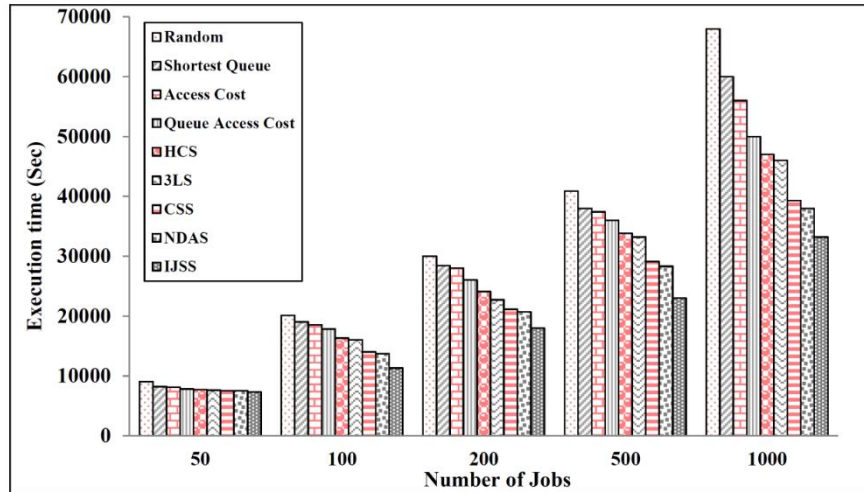**Fig. 6. Queue time versus number of jobs.**

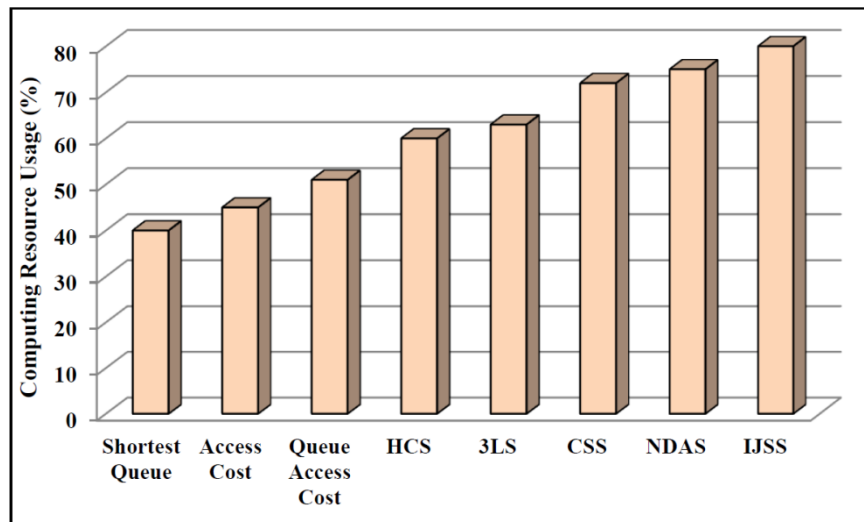**Fig. 7. Execution time versus number of jobs.**

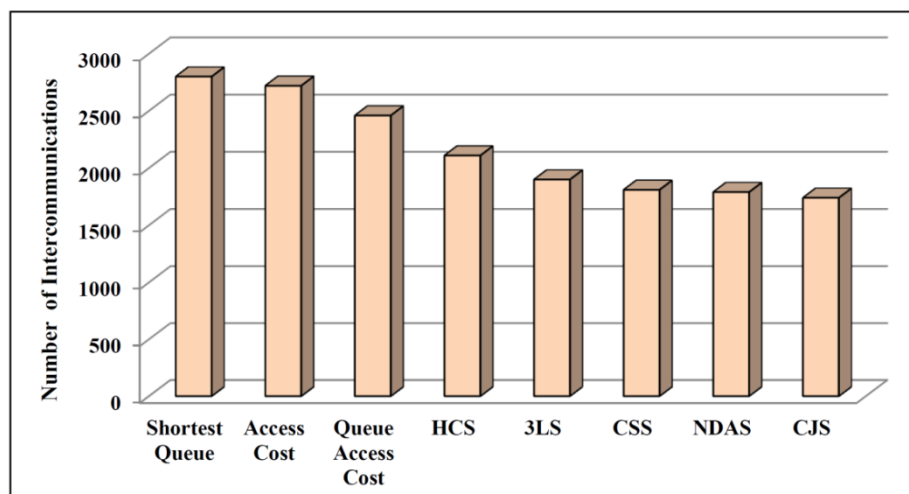**Fig. 8. Computing resource usage for various job scheduling.**

Fig. 9. Number of inter-communications

7. CONCLUSION

Data Grid has evolved to be the solution for data-intensive applications, such as astrophysics, and computational genomics. Yet effective scheduling in data grid environments is challenging, due to a need to address a variety of metrics and constraints (e.g., resource utilization, response time,) while dealing with multiple, potentially independent sources of jobs and a large number of storage, compute, and network resources. Considering various requirements of jobs during scheduling decision within Grid environments is the main concern of this paper. The scheduler can make "intelligent" decisions by taking into account the changing state of the network, the locality and the size of the data and the computational power. To achieve a more appropriate scheduling in Grids, an algorithm named IJSS was proposed in this paper to discuss the problem of simultaneously considering data-intensive and computation-intensive dimensions of the jobs.

The IJSS strategy takes network characteristics as a primary class criterion in the scheduling decision along with computations and data. It was also deduced that a combination of data transfer cost, network cost and computation cost can considerably optimize the Grid scheduling and execution process which was the key message of the IJSS scheduling approach. A grid simulator (i.e. OptorSim) was utilized to evaluate the IJSS algorithm. The simulation results showed that the new algorithm enhanced the performance of the grid environment and thus decreased the job's average total time. From a simulation perspective, it will be interesting to evaluate the results in more complex networks. Another interesting issue is modeling a real grid scenario with the existing resources and real job traces.

REFERENCES

[1] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," IEEE Transactions on Software Engineering, 15: pp. 427-1436, 1989.

[2] S. Kardani-Moghadam, F. Khodadadi, and R. Entezari-Maleki, A. Movaghar, "A hybrid genetic algorithm and variable neighborhood search for task scheduling problem in grid environment," Procedia Engineering, 29: pp. 3808-3814, 2012.

[3] R. Entezari-Maleki, and A. Movaghar, "A genetic-based scheduling algorithm to minimize the makespan of the grid applications," In: Grid and DistributedComputing Conference, Communications in Computer and Information Science (CCIS), pp. 22-31, 2010.

[4] Z. Mousavinasa, R. Entezari-Maleki, and A. Movaghar, "A bee colony task scheduling algorithm in computational grids," In: Iternational Conference on Digital Information Processing and Communications (ICDIPC), pp. 200-211, 2011.

[5] B. Radha, and V. Sumathy, "Enhancement of grid scheduling using dynamic error detection and fault tolerance," International Journal of Computer Applications, 31(7), 2011.

[6] R. Shakerian, S.H. Kamali, M. Hedayati, and M. Alipour, "comparative study of ant colony optimization and particle swarm optimization for grid scheduling," The Journal of Mathematics and Computer Science, 2 (3): pp. 469-474, 2011.

[7] S.H. Kamali, M. Hedayati, R. Shakerian, and S. Ghasempour, "Using identity-based secret public keys cryptography for heuristic security analyses in grid computing," The Journal of Mathematics and Computer Science, 3 (4): pp. 357-375, 2011.

[8] J. Nabrzyski, J.M. Schopf, and J. Weglarz, Grid Resource Management, Kluwer Publishing, 2003.

[9] L.R Anikode, and B. Tang, "Integrating scheduling and replication in data grids with performance guarantee," In: Global Telecommunications Conference, pp. 1-6, 2011.

[10] J. Basney, M. Livny, and P. Mazzanti, "Utilizing widely distributed computational resources efficiently with execution domains," Comput Phys Commun, 140(1): pp. 246-252, 2001.

[11] J. Zhang, B. Lee, X. Tang, and C. Yeo, "Improving job scheduling performance with parall el access to replicas in data grid environment," J. Supercomput. 56: pp. 245-269, 2011.

[12] G. Falzon, and M. Li, "Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments," J. Supercomput. 59: pp. 104-130, 2012.

[13] S.Abdi, and S. Mohamadi, "Two level job scheduling and data replication in data grid," International Journal of Grid Computing & Applications, 1(1), 2010.

[14] K.Yi, F. Ding, and H. Wang, "Integration of task scheduling with replica placement in data grid for limited disk space of resources," In: Fifth Annual China Grid Conference, pp.37-42, 2010.

[15] A. Jula, N. Khatoon Naseri, and AM. Rahmani, "Gravitational attraction search with virtual mass GASVM to solve static Grid job scheduling problem," The Journal of Mathematics and Computer Science, 1 (4): pp. 305-312, 2010.

[16] A.S Izadi, A.R. Sahab, and J. Vahidi, "A new mechanism for traffic reduction the service resource discovery protocol in ad-hoc grid

network," The Journal of Mathematics and Computer Science, 6 (2): pp. 129-138, 2013.

[17] H.M. Wong, V. Bharadwaj, Y. Dantong, and T.G. Robertazzi, "Data intensive grid scheduling: multiple sources with capacity constraints," In: Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS), pp. 163-170, 2004.

[18] K. Li, Z. Tong, D. Liu, T. Tesfazghi, and X. Liao, "PTS-PGATS based approach for data-intensive scheduling in data grids" Frontiers of Computer Science, 5(4): pp. 513-525, 2011.

[19] W. Liu, R. Kettimuthu, B. Li, and I. Foster, "An adaptive strategy for scheduling data-intensive applications in grid environments" In: 17th international conference on telecommunication, pp. 642-649, 2010.

[20] F. Xhafa, and A. Abraham, "Computational models and heuristic methods for grid scheduling problems," Future Gener Comp Sy. 26: pp. 608-621, 2010.

[21] J.M. Schopf, "Ten actions when grid scheduling the user as a grid scheduler," Chapter 1, 2004.

[22] R.S. Chang, C.Y. Lin, and C.F. Lin, "An adaptive scoring job scheduling algorithm for grid computing" Inform Sciences. 207: pp. 79-89, 2012.

[23] A. Chaudhuri, D. Jana, and B.B. Bhaumik, "Optimal model for scheduling of computational grid entities" In: India Conference (INDICON), pp. 1-6, 2011.

[24] I. Foster, and K. Ranganathan, "Design and evaluation of dynamic replication strategies for high performance data grids," In: Proceedings of International Conference on Computing in High Energy and Nuclear Physics, 2001.

[25] [25] I. Foster, and k. Ranganathan, "Identifying dynamic replication strategies for high performance data grids," In: Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing, pp. 75–86, 2002.

[26] R, Chang, J. Chang, and S. Lin, "Job scheduling and data replication on data grids," Future Gener Comp Sy. 23: pp. 846-860, 2007.

[27] A. Horri, R. Sepahvand, and G.H. Dastghaibyfard, "A hierarchical scheduling and replication strategy," International Journal of Computer Science and Network Security, 8, 2008.

[28] N. Mansouri, G.H. Dastghaibyfard, and E. Mansouri, "Combination of data replication and scheduling algorithm for improving data availability in data grids" J. Netw. Comput. Appl. 36: pp. 711-722, 2013.

[29] J. Zhang, B. Lee, X. Tang, and C. Yeo, "Impact of parallel download on job scheduling in data grid environment," In: Seventh International Conference on Grid and Cooperative Computing, pp. 102-109, 2008.

[30] H.H. Mohamed, and D.J. Epema, "An evaluation of the close-to-files processor and data co-allocation policy in multi-clusters," In: International Conference on Cluster Computing, IEEE Society Press. IEEE Society Press, pp. 287-298, 2004.

[31] M. Tang, B.S. Lee, X. Tang, and C. Yeo, "The impact of data replication on job scheduling performance in the data grid" Future Gener. Comp. Sy. 22: pp. 254-268, 2006.

[32] S. Vazhkudai, "Enabling the co-allocation of grid data transfers," in: Proceedings of the Fourth International Workshop on Grid Computing, pp. 44-51, 2003.

[33] S. Kumar, and N. Kumar, "Network and data location aware job scheduling in grid: improvement to GridWay Meta scheduler," International Journal of Grid and Distributed Computing, 5(1), 2012.

[34] C. Wu, and R. Sun, "An integrated security-aware job scheduling strategy for large-scale computational grids," Future Generation Computer Systems, 26 (2): pp. 198–206, 2010.

[35] M. Hemamalini, and M.V. Srinath, "State of the art: task scheduling algorithms in a heterogeneous grid computing environment," Engineering research and management journal, 1(1): pp. 15-21, 2014.

[36] D.I. George Amalarethinam, and P. Muthulakshmi, "An overview of the scheduling policies and algorithms in grid computing," International Journal of Research and Reviews in Computer Science 2 (2): pp. 280-294, 2011.

[37] D.G Cameron, A.P. Millar, C.C Nicholson, R. Carvajal-Schiaffino, F. Zini, and k. Stockinger, "Optorsim: a simulation tool for scheduling and replica optimization in data grids," In: International conference for computing in high energy and nuclear physics (CHEP'04), 2004.