International Journal of Grid Computing & Applications (IJGCA) Vol.1, No.1, September 2010

TWO LEVEL JOB SCHEDULING AND DATA REPLICATION IN DATA GRID

Somayeh Abdi[1] and Somayeh Mohamadi[2]

[1]Department of Engineering, Eslamabad Gharb branch Islamic Azad University, Eslamabd Gharb, Kermanshah, Iran
somayeh.abdi@gmail.com

[2]Department of Engineering, Ghasre Shirin branch Islamic Azad University, Ghasre Shirin, Kermanshah, Iran
mohamadi.s@gmail.com

ABSTRACT

Data Grid environment is a geographically distributed that deal with date-intensive application in scientific and enterprise computing. In data-intensive applications data transfer is a primary cause of job execution delay. Data access time depends on bandwidth, especially when hierarchy of bandwidth appears in network. Effective job scheduling can reduce data transfer time by considering hierarchy of bandwidth and also dispatching a job to where the needed data are present. Additionally, replication of data from primary repositories to other locations can be an important optimization step to reduce the frequency of remote data access. Objective of dynamic replica strategies is reducing file access time which leads to reducing job runtime. In this paper we develop a job scheduling policy, called TLSS (Two level Scheduling strategy), and a dynamic data replication strategy, called TLRS (Two level Replication Strategy), to improve the data access efficiencies in a cluster grid. We study our approach and evaluate it through simulation. The results show that combination of TLSS and TLRS has improved 17% over other combinations.

KEYWORDS

Grid, Data Grid, Job Scheduling, Data Replication, Simulation

1. INTRODUCTION

In the in creasing demand of scientific and large-scale business application, a large amount of data are generated and spread for using by users around the world. A Grid is a distributed collection of computer and storage resources maintained to serve the needs of some community or virtual organization. Data Grid is an integrating architecture that allow connect a collection of hundreds of geographically distributed computers and storage resources located in different part of the world to facilitate sharing of data and resources [1]. Size of data that needs to be accessed on the Data Grid may be up to petabytes in the near future. Data-intensive jobs are one major kind of jobs in data grid, and their scheduling strategies are regarded as one of the most important research fields [2]. Size of data that needs to be accessed on the Data Grid is in terabytes or petabytes in the near future that leads to many challenges in Data Grid. One of the aspects that must be considered is how the scheduling efficiently work with the amount of data

need for each job and the impact of replication mechanism to the scheduling performance [1]. In data-intensive applications, the locations of data required by the job impact the Grid scheduling decision and performance greatly [3].
As a data grid represents a distributed storage solution, its performance is dictated by the access latency and bandwidth of the underlying communication network. In large-scale data-intensive applications, data transfer time is the primary cause of job execution delay. To mitigate the effect of such problems and facilitate efficient system performance, replication strategies and algorithms are employed [4]. The replication mechanism determines which file should be replicated, when to create new replicas and where the new replicas should be placed. Replication methods can be classified as static and dynamic. For the static replication methods, a replica can persist until it is deleted by users or its duration is expired. The drawback of static replication is clear; when client access pattern changes, the benefits brought by replica will decrease. On the contrary, dynamic replication takes into consideration the changes of the Grid environment, and it automatically creates new replicas for popular data files or moves the replicas to other sites when is needed to improve the performance [3]. When users' jobs access a large amount of data from remote sites, dynamic replica optimizer running in the site tries to store replicas on local storage for future possible repeated requests. If most data resides on the same site where they are needed, the frequency of remote data access is going to decrease. This can reduce job execution time and increase the robustness of grid application.

In this paper we develop new scheduling and replication algorithms that reduce data transfer time and job execution time. Our new scheduling policy considers the locations of required data, hierarchy of bandwidth and the capacity of computing nodes. It is called TLSS (Two Level Scheduling Strategy). TLSS uses hierarchical scheduling and takes into consideration the cluster information to reduce the data transfer time by reducing number of intercommunications. When data have to be replicated, we develop a replication strategy, called TLRS (Two Level Replication Strategy). It considers bandwidth as an important factor for replica selection and replica placement. It also increases the chances of accessing data at a nearby node in cluster grid.

The rest of the paper is organized as follows, in section 2 we present a summary of existing and related work. In section 3, a two layered hierarchical structure is proposed for data grid based on classification of networks, along with a scheduling algorithm and replication algorithm for this structure, Section 4 describes our experiments and the results achieved followed by conclusion in section 5.

2. RELATED WORK

There are some recent works that address the problem of scheduling and/ or replication in Data Grid as well as the combination between them.
In [3], it considers two centralized and decentralized replication algorithms. In centralized method, replica master uses a table that ranks each file access in descending order. If a file access is less than the average, it will be removed from the table. Then it pop files from top and replicates using a response-time oriented replica placement algorithm. In the decentralized method, every site records file access in its table and exchange this table with neighbours. Since every domain knows average number of access for each file and then deletes those files whose access is less than the average, and replicates other files in its local storage.
In [4], an algorithm for a 2-level hierarchical structure based on internet hierarchy (BHR) has been introduced which only considers dynamic replication and does not consider scheduling. Nodes in the first level are connected to each other with high speed networks and in the second level via internet. The algorithm replicates the file to the site if there is enough space. Next it, accesses the file remotely if the file is available in the sites that are in the same region. Otherwise it tries to make available space by deleting files using LRU (Least Recently Used)

method, and replicates the file. It assumes that master site always has a safe copy of file before deleting.

In [5], an algorithm for a two-level hierarchical structure based on internet hierarchy (BHR) has been introduced which only considers dynamic replication and does not consider scheduling. Nodes in the first level are connected to each other with high speed networks and in the second level via internet. The algorithm replicates the file to the site if there is enough space. Next it, accesses the file remotely if the file is available in the sites that are in the same region. Otherwise it tries to make available space by deleting files using LRU (Least Recently Used) method, and replicates the file. It assumes that master site always has a safe copy of file before deleting.

In [6], a structure with few networks connected via internet has been presented and an algorithm similar to [5], along with scheduling is proposed. For replicating a file, first computes the total transfer time, then it selects the best node with shortest transfer time.

In [7] authors introduce dynamic replication placement (RP) that categorizes the data based on their property. This category is used for job scheduling and replication. Then a job is allocated to a site which has the file in the required category, this leads to reduce the cost for file transfer.

In [8] a Genetic Algorithm based co-scheduling of data and jobs of independent nature was proposed; the GA is executed to converge to a schedule by looking at the jobs in the scheduler queue as well as the replicated data objects at once. A performance overhead might be incurred by the system as a result of delaying the replication of the data until the scheduling time. The authors also use an objective function that assumes infinite availability of storage for all data objects which is infeasible in a realistic grid setting.

In [19,20], Ranganathan and Foster present six different replica strategies: (1) No replication or caching, (2) Best Client: a replica is created at the best client that has the largest number of requests for the file, (3) Cascading replication: once popularity exceeds the threshold for a file at a given time interval, a replica is created at next level which is on the path to the best client, (4) Plain caching: the client that requests the file stores a copy of the file locally, (5) Caching plus Cascading Replication: this combines Plain caching and Cascading replication strategy, and (6) Fast Spread: replicas of the file are created at each node along its path to the client. These strategies are evaluated with three different data patterns: (1) Random access: there is no locality in access patterns, (2) data access with a small degree of temporal locality (recently accessed file are likely to be accessed again), (3) Data access with a small degree of temporal and geographical locality. (Files recently accessed by a site are likely to be accessed by nearby site.) The results of simulations indicate that different access pattern needs different replica strategies. With suitable strategies, they can dramatically improve the performances in bandwidth savings or access latency. Two strategies performed the best in the simulations: Cascading and Fast Spread when compared to traditional strategies. Ranganathan and Foster also propose a variety of techniques to intelligently replicate data across sites and assign jobs to sites in data grid [21]. They have conducted a study of the performances of various scheduling algorithms by using simulator. A scheduler selects a remote site to dispatch a job based on one of the four algorithms: (1) JobRadom: schedule a job randomly, (2) JobLeastLoaded: schedule a job to where there is the least number of jobs waiting to run, (3) JobDataPresent: schedule a job to where it has the least load and requested data, and (4) JobLocally: always run jobs locally. These job scheduling algorithms are combined with three different replication strategies: (1) DataDoNothing: there is no replication, (2) DataRandom: once the threshold for a file is exceeded, a replica is created at a random site, (3) DataLeastLoad: once the threshold for a file is exceeded, a replica is created at a site where the least number of jobs are waiting in the queue. The simulations show that loosely coupled jobs and remotely distributed large data sets can be optimized separately. They recognize the significance of data location in job dispatching and scheduling in grids. However, this work only considers jobs that use a single input file and assumes homogeneous sites with a simplified First-In–First-Out (FIFO) strategy within local schedulers.

3. THE PROPOSED METHOD

In this section, we present hierarchical network structure, called cluster grid, and then we proposed two strategies for job scheduling and data replication by considering the hierarchical network structure.

3.1. Network Structure

The Data Grid architecture supporting data replication and job scheduling, called cluster grid, is shown in Figure 1. A cluster represents an organization unit which is a group of sites that are geographically close to each other, each cluster comprises the computers which are connected by a high bandwidth. We define two kinds of communications between sites in a cluster grid, Inter-communication and intra-communication. Intra-communication is the communication between sites within the same cluster and inter-communication is the communication between sites across clusters. Network bandwidth between sites within a cluster will be larger than across clusters. In communication networks, the performance of system is underlying available network bandwidth and data access latency, especially in networks that hierarchy of bandwidth appears. Therefore, to reduce access latency and to avoid WAN bandwidth bottleneck in a cluster grid, it is important to reduce the number of inter-communications. Data Grid Information Service providing resource registration services and keeping track of a list of resources available in the Data Grid. The Grid Scheduler can query this for resource contact, configuration, and status information; Resource discovery identify resources that can be used with their capability through Data Grid Information Service. In this structure we apply distributed dynamic data replication infrastructure. Replica Manager at each site manages the data movement between sites and we add Data Scheduler into this structure as part of Replica Manager. When a job is assigned to LS[1], the DS[2] will responsible for all the data requests by the LS. This data request is generated as soon as job is scheduled into LS queue.

[1] . Local Scheduler

[2] . Data scheduler

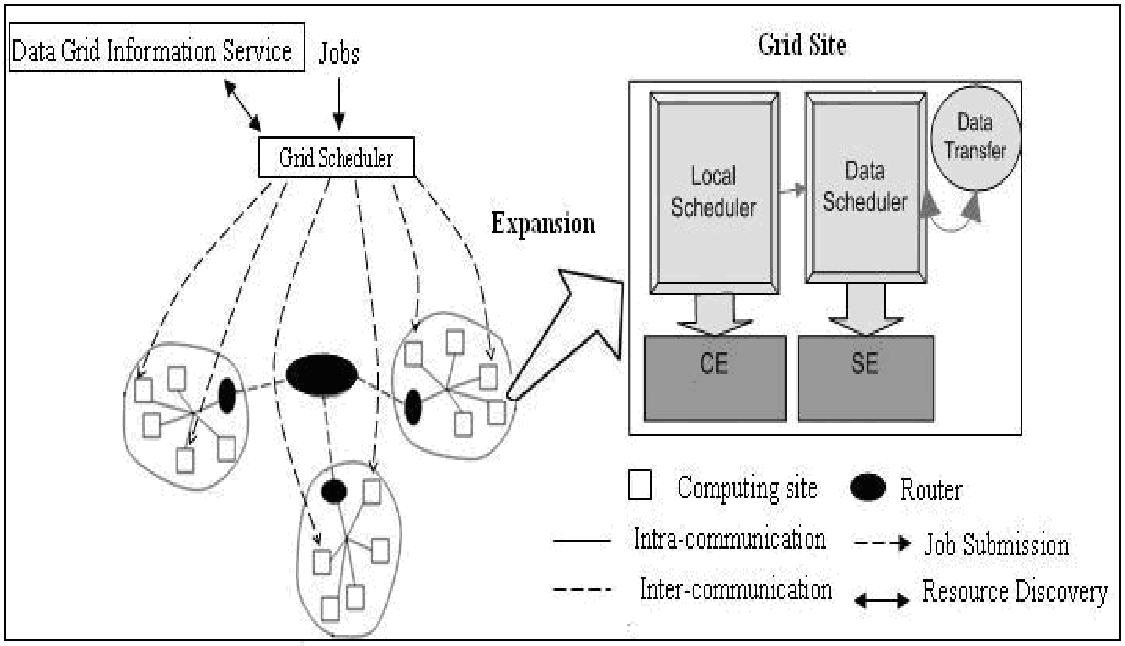


Figure 1. Data Grid Architecture.

3.2. Two level Scheduling Algorithm

In data-intensive applications, data transfer time is the primary cause of job execution delay, the locations of data required by the job impact the Grid scheduling decision and performance greatly.
TLSS (Two level scheduling strategy) considers the locations of required data at cluster level and site level in job scheduling decision. The algorithm determines the best cluster and site respectively and then submit job to LS. A best cluster (and site) is a cluster that holds most of the requested files (from size point of view). This will significantly reduce total transfer time, and reduce the number of inter-communications. We assume that $S_{S,c}$ is total size of the requested files available in site S and cluster C. Also, we assume C_C is total size of the requested files available in cluster C and the needed data to execute job j are represented as: Rj = {LFN1, LFN2,…., LFNn} and $|LFN_i|$ presents size of file LFNi . Additionally we define RelativeLoadi that presents the relative load of site i (based on number of jobs at site i and computing capacity of site i).

$$S_{S,C} = \sum_{\text{for all available LFNi in site S}} |LFN_i| \tag{1}$$

$$C_C = \sum_{S=1}^{N} S_{S,C} \tag{2}$$

$$\text{RelativeLoad}_i = \text{SizeofJobs}_i / C_i \qquad (3)$$

Where N is number of sites in cluster C, Ci is computing capacity (in MIPS3) of site i and SizeofJobsi is the lengths of queued jobs (in MIPS) on site i.

The TLSS can be summarized as follow:

1- Compute C_C for each cluster in data grid from (2)
2- Select the best cluster $C_{\max} = \underset{i=1}{\overset{q}{MAX}}\ C_i$; where q is the number of cluster (i.e. cluster with largest available requested data files)
3- for each site in $C_{\max}$ compute $S_{S,\max}$ from (1)
4- Select the best site $S_{\max,\max} = \underset{k=1}{\overset{r}{Max}} S_{k,\max}$; where r is the number of sites in cluster $C_{\max}$ (i.e. Site with largest available requested data files from size point of view in cluster $C_{\max}$).

5- If there are several sites with most available data in the $C_{\max}$ select the site with minimum relative load from (3).

When a job is assigned to LS, the DS will be responsible for transferring all the data required by the job that aren't exists in local site. The objective of this DS is to transferring the data required by the job to the local site before job execution and it leads to reduction of job execution time. The proposed scheduling algorithm by using the DS and overlapping between queue waiting time and data transferring time, improved the job execution time. Additionally, where there are the same data requirements in several sites, TLSS takes into consideration the computing capacity of sites with a view to reducing the queue waiting time.

3.3. Two level Replication Strategy

After a job is scheduled to Sj , the requested data will be transferred to Sj to become replicas. TLRS (two level replication Strategy) determines wich replica will be transferred to Sj and how to handle this new replica, as shown in Figure 2. TLRS considers the inter-communication bandwidth as the main factor for replica selection / deletion.
For each needed file for job executing, replica manager controls the existence of the file in local site; if file doesn't exist in the local site TLRS first searches the file in local cluster. If the file duplicated in the same cluster, then it will creates a list of candidate replicas and selects a replica with the maximum bandwidth available for transferring it. If there is enough space for new replica, then the new replica will be stored in local site, otherwise it is only stored in the temporary buffer and will be deleted after the job completes.

[3]. Million Instruction Per Second

```
If (the needed replica is not in a site)
{
  Search replica in the same cluster;
    If (replica is in the same cluster)
      Create list of candidate replicas in the same cluster
 else {
     Search replica in other clusters
     Create list of candidate replicas in other cluster;
  }
  Select the replica with most available bandwidth in
  Candidate replica list
  If (there is enough space to Store the new replica)
 {Store it; Exit;   }
  else {
       If (replica is in the same cluster)
      {   Exit; // avoid duplication in the same cluster;    }
        else {
               While (! enough space)
               Use LRU replacement algorithm to Delete
              Replica in local storage that duplicate in the same cluster;
              If (enough space)   {store it; Exit ;}
             else {
                    While (! enough space)
                    Use LRU replacement algorithm to Delete
                    Replica in local storage that duplicate in the other cluster;
                    If (enough space) {store it; Exit ;}
             }
      }
 }
```

Figure 2. Two Level Replication Strategy.

If the file doesn't exit in the same cluster, then TLRS creates a list of replicas in other clusters and selects the replica with the maximum bandwidth available for transferring it. If there is enough space for new replica, then it will be stored in local site, otherwise occupided space will be released to make engough room for this new replica. First, it removes the replicas that already exist in other sites in the same cluster based on LRU(least recently used) replacement algorithm. After all these replicas are deleted, if the space is still insufficient, the TLRS uses LRU replacement algorithm to delete replica in local storage which is duplicated in the other cluster , till it has enough room for the new replica. To conclude, TLRS considers inter-cluster replica transference as very costly. Therefore, the successfully received replicas must be stored locally such that all other sites in the same cluster will access replica with intra-communication way.

4. SIMULATIONS

Gridsim is used as the simulation tool to evaluate the performance of the proposed replication and scheduling algorithms. The Java-based GridSim discrete event simulation toolkit provides Java classes that represent entities essential for application, resource modelling, scheduling of jobs to resources, and their execution along with management [9]. Its Java-based design makes it portable and available on all computational platforms. The components of Gridsim are as follow and also depicted in Figure 3.

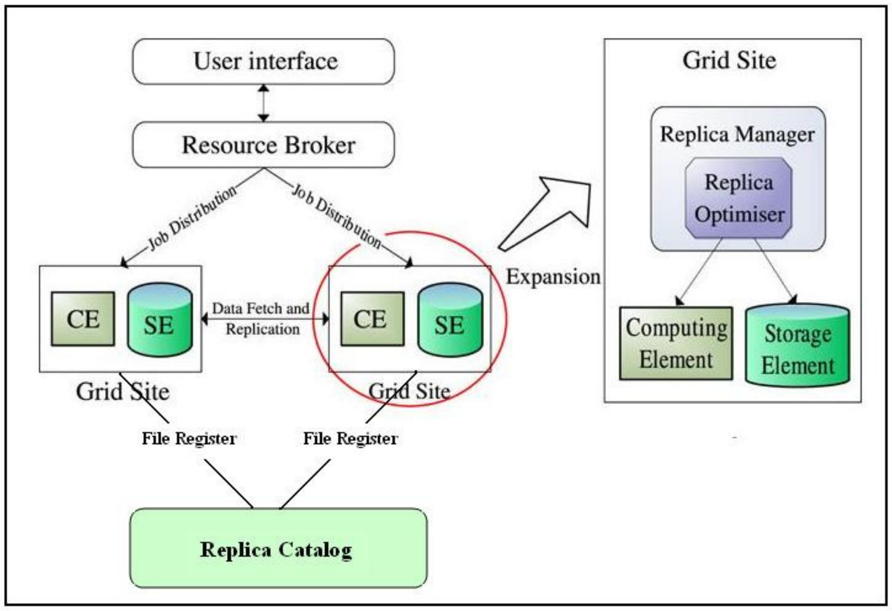


Figure 3. Simulator architecture

Resource Broker: It receives jobs from user and sends them to the best node according to proposed algorithm.
Storage Element (SE): Storage resource in grid.
Computing Element (CE): Computing resource in grid.
Replica Catalogue (RC): This component acts as a centralized RC. It is responsible for indexing available files on the resources and handles queries from users and resources about location of replicas. When each site store a new replica, send a file register request to RC and then RC add this site to the list of sites that holds the replica.
Replica Manager: It controls data transferring in each node and provide a mechanism for accessing the Replica Catalogue.
Replica Optimizer: It has replica algorithm and control file replication according to proposed algorithm.
Based on the scheduling algorithm the broker sends jobs to a node. Each job needs a list of files to run. Reducing file access time is the final objective of optimization algorithms.

4.1. Simulation Environment

There are four clusters in our configuration and each cluster has an average of 13 sites, which all have CE with associated SE. Table1 specifies the simulation parameters used in our study. There are 5 job types; each job type requires 12 files to execute. While running, jobs were randomly picked from 5 job types, then submitted to the Resource Broker. Files are accessed sequentially within a job without any access pattern. To simplify the requirements, data replication approaches in Data Grid environments commonly assume that the data is read-only.

TABLE 1. Simulation parameters

Topology Parameters	**value**
Number of cluster	4
Number of sites in each cluster	13
Storage space at each site	10 GB
Connectivity bandwidth(WAN)	10 Mbps
Connectivity bandwidth(LAN)	1000Mbps
Job parameters	**value**
Number of jobs	500
Number of job types	5
Number of file accessed per job	12
Size of single file	500 MB
Total size of files	50GB

4.2. Simulation results and discussion

TLRS will be compared with LRU (Least Recently Used) and BHR (Bandwidth Hierarchy based Replication). The LRU algorithm always replicates and then deletes those files that have been used least recently. Figure 4 shows the Average job time based on changing number of jobs for 3 algorithms. Figure 5 shows the Average job time for 1000 jobs by 3 mentioned algorithms. TLRS replication strategy uses the concept of “network locality” as BHR [5]. The difference between TLRS and BHR is that required replica within the same cluster is always the top priority used in TLRS, while BHR searches all sites to find the best replica and has no distinction between intra-cluster and inter-cluster. It could be anticipated that TLRS will avoid inter-cluster-communications and be stable in hierarchical network architecture with variable bandwidth. Our method takes benefit from network level locality of BHR. Thus, total job execution time is about 12% faster using TLRS optimizer than BHR.

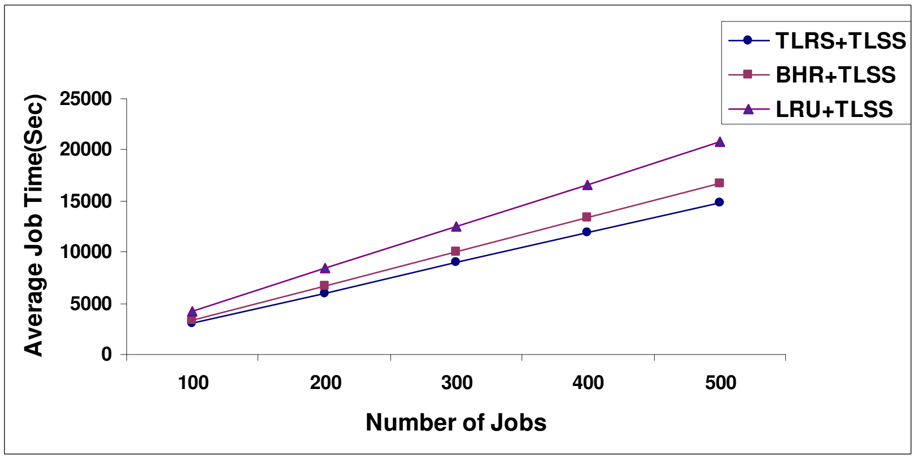


Figure 4. Average job Time based on varying number of jobs.

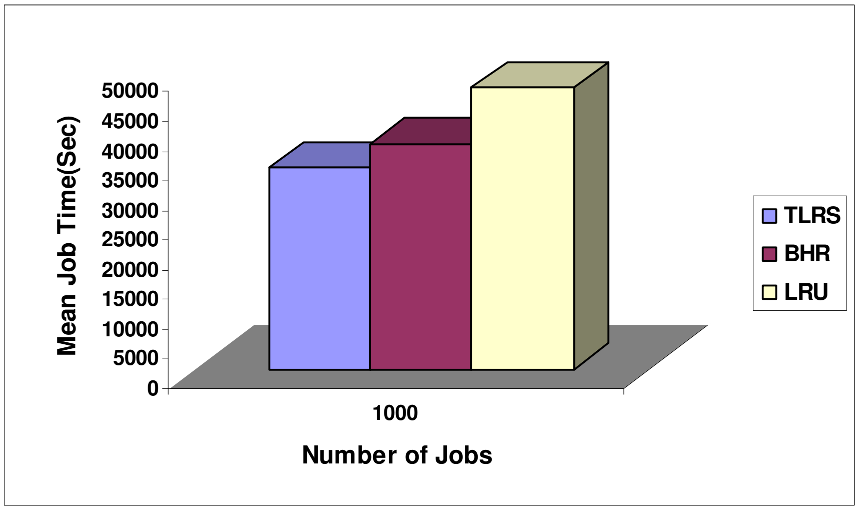


Figure 5. Average Job Time for 1000 jobs.

In the TLSS the job execution time is the Max {file transmission time, queue time} plus job processing time. TLSS will be compared with DPRL and Relative Load scheduling strategies. DPRL (Data Present Relative Load) searches all sites to find available CE by using a combination of Data present for the files and the Relative load of sites. Figure 6 shows the Average job time based on changing number of jobs for 3 algorithms. Since TLSS schedules jobs to certain specific cluster and specific sites according to requested data files. Therefore, jobs would be executed on a cluster with the most needed files. Since the file transmission time is the most important factor to influence the job execution time for data-intensive jobs in data grids, TLSS with TLRS can reduce the file transmission time effectively by virtue of valid scheduling and proper data replication.

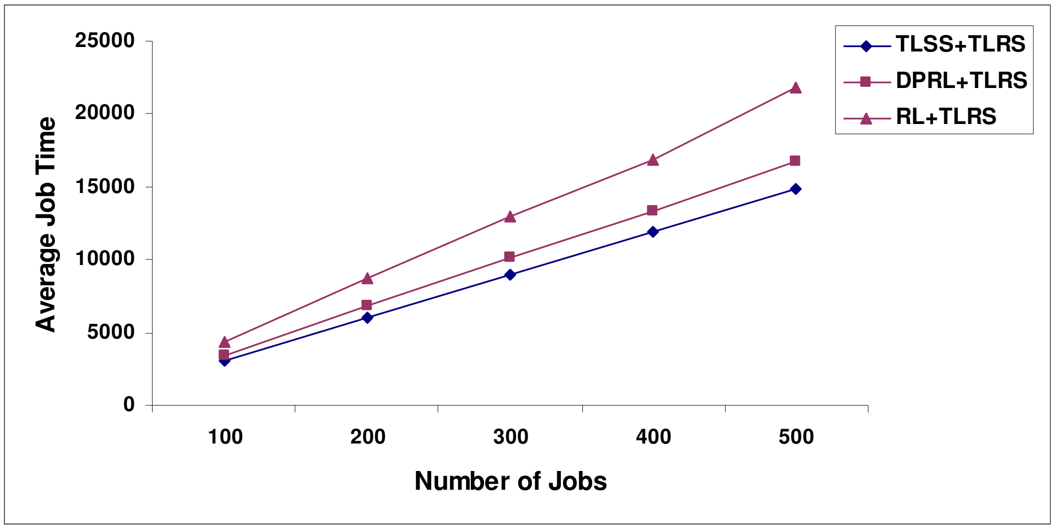
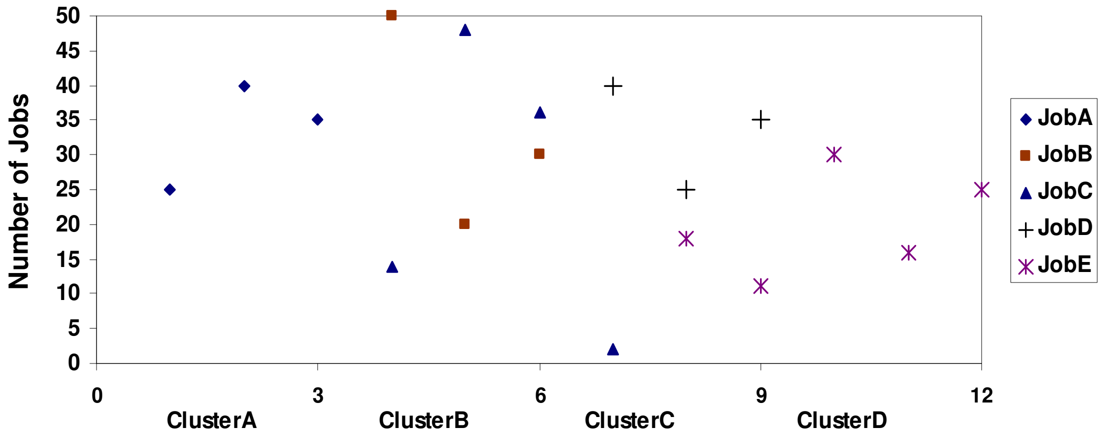


Figure 6. Average job Time based on varying number of jobs.

To analyze the distribution of jobs, we run a simulation where there is a grid system with four clusters. Each cluster has three grid sites and 500 jobs. Figure 7 shows the distribution of where jobs are executed. Since TLSS schedules jobs to certain specific sites and specific cluster according to requested data files.Therefore, jobs would be executed on a cluster with the most needed files. It can be observed that the same type of jobs is almost executed at the same cluster as shown in Figure 7(a). On the contrary, DPRL does not take into cosideration cluster information and it only schedules jobs to certain specific site, therefore different job types would be executed on a cluster and the number of inter-communication would be increased. It will lead to more overhead in transferring file replicas.The job distribution of DPRL is shown in Figure 7(b) .



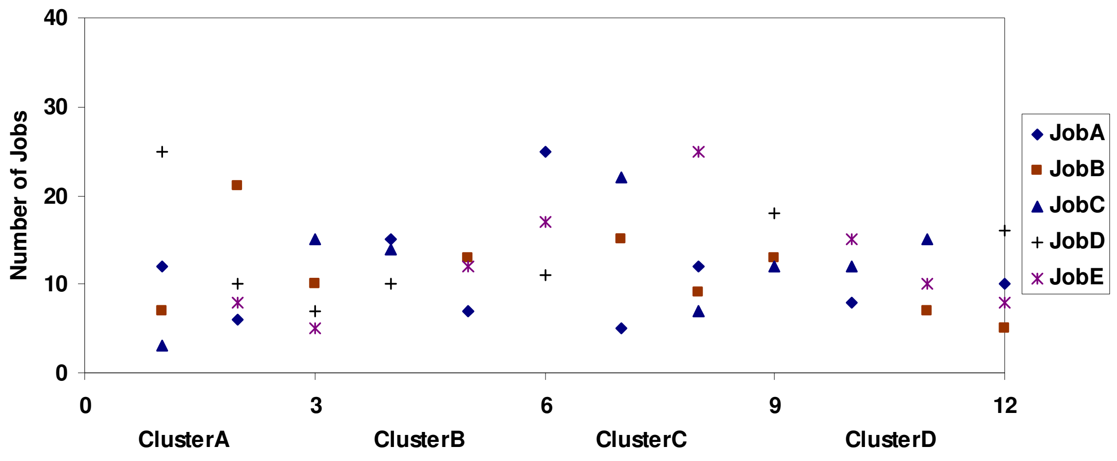


Figure 7. 500 job distribution (a) TLSS with TLRS (b) DPRL with LRU.

The average number of inter-communications for a job execution is illustrated in Figure 8. By selecting the best cluster and best site, TLSS with TLRS can decrease the number of inter-communications effectively. Overall the simulation results with Gridsim show better performance (over 12%) comparing to current algorithms.

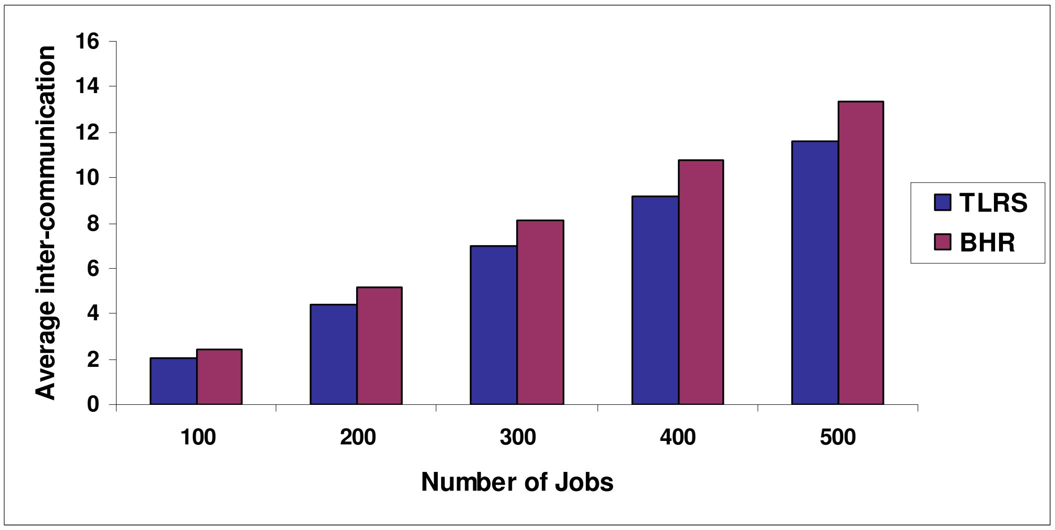


Figure 8. Average number of inter-communications

Figures 9 shows the average job time for 500 jobs. We compare TLRS, BHR and LRU algorithms for varying inter-communication bandwidth.

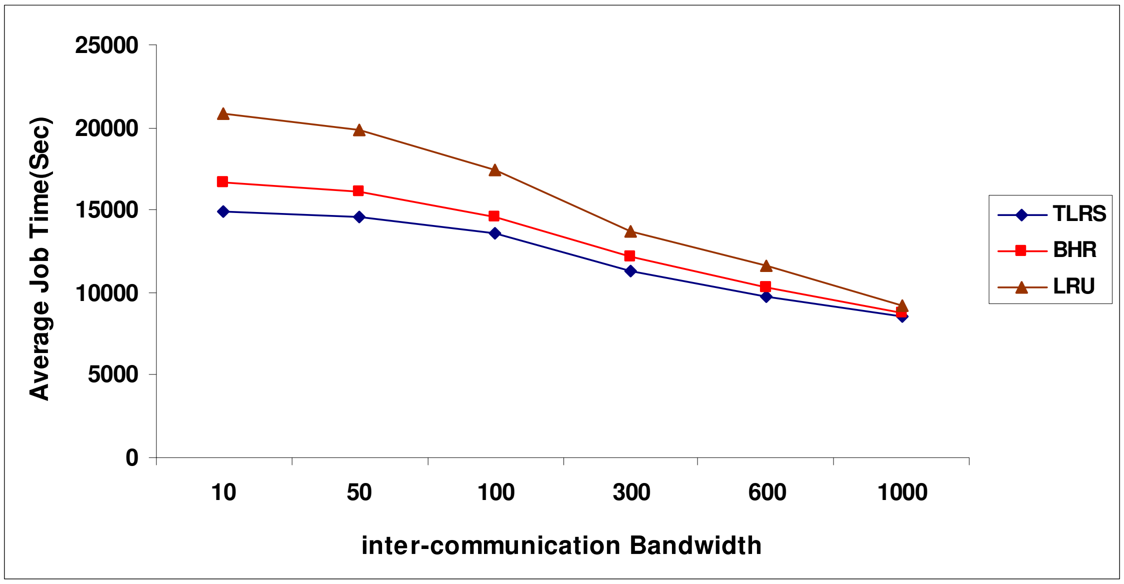


Figure 9. Average Jobs Time with varying inter-communication bandwidth for 500 jobs

As inter-communication bandwidth increase 3 mentioned algorithms will converge. We can conclude that TLRS strategy can be effectively utilized when hierarchy of bandwidth appears.

5. Conclusion and Future Work

In this paper a two layered hierarchical structure for dynamic replicating file and cluster scheduling in data grids was proposed. To achieve good network bandwidth utilization and reduce data access time, we propose a job scheduling policy (TLSS) that considers not only computational capability, job type and data location but also it considers cluster information in job placement decision. We study and evaluate the performance of various replica strategies and different scheduling algorithm combinations. The simulation results show, first of all, that TLSS and TLRS both get better performances than other scheduling policy and replica strategies. Second, we can achieve particularly good performance with TLSS where jobs are always scheduled to cluster with most of the needed data, and a separate TLRS process at each site for replication management. Experimental data shows TLSS scheduling with TLRS replica strategy outperforms others combinations in total job execution time.

References

[1] Jianhua Jiang, Huifang Ji, “scheduling algorithm with potential behaviors”, Journal of Computers, VOL. 3 , NO. 12 , December 2008.

[2] Ming Tang, Bu-Sung Lee, Xueyan Tang, Chai-Kiat Yeo .” The Impact of Data Replication on Job Scheduling Performance in the Data Grid”, Future Generation Computer Systems, Volume 22, Issue 3, February 2006, Pages 254-268

[3] Ali Elghirani, Riky Subrata, Albert Y. Zomaya, and Ali Al Mazari., “Performance Enhancement through Hybrid Replication and Genetic Algorithm Co-Scheduling in Data Grids”, Advanced Networks Research Group, School of Information Technologies, University of Sydney, NSW 2006 Australia.

[4] Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko: "Dynamic Grid Replication Strategy based on Internet Hierarchy", Book Series Lecture Notes in Computer Science, Grid and Cooperative omputing book,Publisher Springer, August 2005, Volume 3033/2004, Pages 838-846

[5] Ruay-Shiung Chang, Jih-Sheng Chang, Shin-Yi Lin, "Job scheduling and data replication on data grids", Future Generation Computer Systems, Volume 23, Issue 7, August 2007, Pages 846-860

[6] Nhan Nguyen Dang, Sang Boem Lim2: "Combination of Replication and Scheduling in Data Grids", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.3, March 2007.

[7] T. Phan, K. Ranganathan, and R. Sion, "Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm", Job scheduling strategies for parallel processing (11th international workshop), JSSPP 2005, Cambridge MA, 2005.

[8] Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", SOFTWARE—PRACTICE AND EXPERIENCE Softw. Pract. Exper. 2002; 32:135–164 (DOI: 10.1002/spe.432).

[9] Klaus Krauter and Manzur Murshed, GridSim a toolkit for the modelling and simulation of distributed resource management and scheduling for Grid computing, CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2002; 14:1175–1220 (DOI: 10.1002/cpe.710).

[10] The Data Grid Project. http://www.eu-datagrid.org

[11] Parallel workload Project. http://www.parrallelworkload.org.

[12] The European data grid project.

[13] W.H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in OptorSim, in: Proceedings of the Third ACM/IEEE International Workshop on Grid Computing, Grid2002, Baltimore, USA, in: Lecture Notes in Computer Science, vol. 2536, 2002, pp. 46–57.

[14] E. Deelman, H. Lamehamedi, B. Szymanski, S. Zujun, Data replication strategies in grid environments, in: Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing, ICA3PP'2002, IEEE Computer Science Press, Bejing, China, 2002, pp. 378–383.

[15] H.H. Mohamed, D.H.J. Epema, An evaluation of the close-to-files processor and data co-allocation policy in multiclusters, in: 2004 IEEE International Conference on Cluster Computing, IEEE Society Press, San Diego, California, USA, 2004, pp. 287–298.

[16] M. Carman, F. Zini, L. Serafini, K. Stockinger, Towards an economybased optimisation of file access and replication on a data grid, in: Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2002, IEEE-CS Press, Berlin, Germany, 2002, pp. 340–345.

[17] P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, Advanced replica management with reptor, in: Proceedings of 5th International Conference on Parallel Processing and Applied Mathemetics, PPAM 2003, Czestochowa, Poland, September 2003, pp. 848–855.

[18] D.G. Cameron, A.P. Millar, C. Nicholson, OptorSim: A simulation tool for scheduling and replica optimisation in data grids, in: Proceedings of Computing in High Energy Physics, CHEP 2004, Interlaken, Switzerland, September 2004.

[19] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid". In Proceedings of the International Grid Computing Workshop, Denver, Colorado, USA, 2001.

[20] I. Foster, 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, Beijing, China, September 2001.

[21] I. Foster, K. Ranganathan, Decoupling computation and data scheduling in distributed data-intensive applications, in: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, IEEE, CS Press, Edinburgh, UK, 2002, pp. 352–358