# Data Replication and Job Scheduling on Cluster Grid for Data-Intensive Applications

*Ruay-Shiung Chang, Shin-Yi Lin, and Jih-Sheng Chang*
Department of Computer Science and Information Engineering, National Dong Hwa University, Shoufeng, Hualien 974, TAIWAN
rschang@mail.ndhu.edu.tw

**Abstract**

In data grids, distributed scientific and engineering applications often require access to large amount of data (terabytes or petabytes). Data access time depends on bandwidth, especially in cluster grid. Network bandwidth within a grid cluster is larger than across clusters. In a communications environment, the major bottleneck to support fast data access in Grids is the high latencies of Wide Area Networks (WANs). Effective scheduling in such network architecture can reduce the amount data transfer across WANs by dispatch a job to where the needed data is present. In another alternative, data replication mechanism generates multiple copies of the existing data to reduce the access opportunity from remote site. To avoid WAN bandwidth bottleneck in a cluster grid, we develop a job scheduling policy, called HCS (Hierarchical Cluster Scheduling), and dynamic data replication strategy, called HRS (hierarchical Replication Strategy) to improve the latency of data accesses. We use simulation studies to evaluate various data access situations. The simulation results show that HCS and HRS successfully reduces data access time and the amount of inter-communications in comparing to other methods in cluster grid.

**Keywords: Cluster Grid, Job Scheduling, Data Replication**

## 1. Introduction

In data grid [1], Distributed scientific and engineering applications often require access to large amount of data (terabytes or petabytes). Access distributed and huge amount of data depend on network bandwidth, the broader bandwidth you take, the less access latency you get. The major bottleneck for supporting fast data access in Grid is the high latencies of WANs and Internet. Namely, slow data access can throttle the performance of data-intensive applications running on grid computers. This situation can observe in hierarchical network structure. In Figure1.1, a simplest hierarchical form of a grid system, called cluster grid, provides a computational service to the group level. A Cluster represents organization unit which is group of sites that are geographically located closely over Internet. We define two kinds of communications between sites in cluster grid. Intra-communication is the communication between sites within cluster. On the contrary, inter-communication is the communication between sites across clusters. Network bandwidth between sites within a cluster will be broader than across clusters. When the required data is intra-communication, less time will be consumed to fetch it. Therefore, to reduce access latency and to avoid WAN bandwidth bottleneck in cluster grid, it is important to keep away from large number of inter-communications.

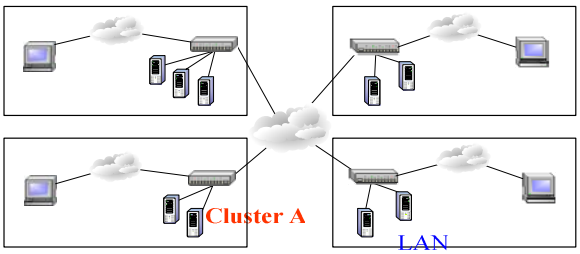

Figure 1.1: The Cluster Architecture

To address this problem, we make inter-communication into two aspects, job scheduling and replication mechanism. Consider a case that any of the authorized users submits jobs to solve data-intensive problems. We demand that jobs be executed as fast as possible. The size of the data used on Data Grid is from terabyte to petabyte. Scheduling is considered a necessity, because data movement is time-consuming. Making scheduling decisions for each job on the appropriate resources is important based on workload and features of computational capability, location of data, and network load. If scheduling a job to a site where the required data is present, the job can process data in this site without any transmission delay for getting data from a remote site. Otherwise, accessing data will have long latency and consume network bandwidth. It will also encumber the efficiency of job execution. By data-intensive nature, it is important to take data location into account when dispatching job to a suitable site.

Therefore, it would be inefficient that scheduling algorithm invariably stresses the importance of computational capability and disregard data locations, and vice versa. Insufficient computing capability and cost for fetching remote data both encumber the efficiency of job execution. Job scheduling policy affects implicitly the chances of accessing data from remote sites. If we consider additional cluster location, inter-communications could be avoided.

Data replication is an important optimization step to manage large data by replicating data in geographically distributed data stores. Previous replication strategies show that replicating data can offer high data availability, low bandwidth consumption. If a site increases hit ratio of amount of data, the frequency of remote data access is going to decrease. This can reduce job execution time and increase the robustness of Grid application. Inter-communications also can be avoided, if data within cluster is the top priority for accessing.

Based on the above-mentioned principles, our scheduling policy considers the locations of required data, the access cost and the job queue length of a computing node, called HCS (Hierarchical Cluster-based Scheduling). HCS is a hierarchical scheduling model that takes cluster information account and reduces search time of appropriate computing node. Our replication strategy, called HRS (Hierarchical Replication Strategy), integrates previous replication strategy and increases the chances of accessing data at a nearby node. They are simulated in OptorSim[2] and experiments with various replica strategies. The result show that HCS and HRS successfully reduces data access time and the amount of inter-communications in comparing to other combinations in cluster grid.

This rest of this thesis is organized as follows. Section 2 gives an overview of previous work. Section 3 introduces our HCS policy and HRS replication strategy. We show results from simulations in Section 4. Finally, Section 5 concludes the thesis and outlines some future work.

## 2. Related Work

As jobs are data intensive, scheduling issues often involve effective computation and data management in the Data Grids. The replication of data sets is not a really new technique. Data replication has been around for decades and it is now adapted to the Grid environment. Similarly, the scheduling on the data grid is still a recent grid computing activities. In [3], K. Ranganathan and I. Foster present six different replica strategies and evaluate with three different data

patterns. The results of simulation indicate that different access pattern needs different replica strategy and they have dramatically improved in bandwidth savings or access latency. Two strategies performed the best in their simulation: Cascading and Fast Spread when compared to traditional strategies. They only show that the replica strategy to be used depending on the data access patterns.

K. Ranganathan and I. Foster also propose a variety of techniques to intelligently replicate data across sites and assign jobs to sites in data grid [4]. They show that job scheduling involves loosely coupled jobs and large data sets distributed remotely, so that these two methods can be implemented and optimized separately. The significance of data location also has recognized in grid scheduling.

These replication strategies mentioned above [3][4] aimed to reduce the message traffic in the network but the data mapping is not optimal. In [5], a replication algorithm is tested which uses a cost model to predict whether replicas are worth creating. It is found to be more effective in reducing average job time than the base case where there is no replication. The simulation architecture used was based on a structure that contrasts to the Data Grid architecture.

Like previous scheduling algorithm, the Close-to-Files (CF) algorithm [6] schedules a job to least load processors close to a site where data are present. The simulation results show that CF can achieve good performance when compared to WF (Worst-Fit) job placement algorithm, which places job components on the execution sites with the largest number of idle processors.

A closer research to the results presented in our paper is BHR (Bandwidth Hierarchy based Replication) [7] and [4]. BHR extends current site-level replica optimization study into the “network-level” based on hierarchy of bandwidth appeared in Internet. It means that BHR maximizes the number of required data in the same region in order to fetch replica faster, since bandwidth within region would be broader. They record regional popularity of files. BHR optimizer selects the best replica for a job and if local storage is already fill up, it will delete duplicated replica in other site within region. In case of storage space is still deficient, BHR remove unpopular files from the “regional point” for the second time. Our replication strategy, HRS, follow BHR concept of maximizing hit ratio of required data within a cluster. Unlike BHR, ours is focus on avoiding large number of inter-communications.

## 3. Scheduling and Replication Algorithm

We propose two functionalities by utilizing the hierarchical network structure: HCS, which is a job scheduling policy, and HRS, which is a replication strategy. For each, we define and evaluate various different algorithms by using a simple network performance model.

### 3.1. Monitoring Network Performance

A poor network performance will limit the efficiency of data transfer and increase the job execution time further. Thus, network performance is the main criteria used in evaluating the access cost of required files and replica selection. However, network performance has kaleidoscopic changes. Predicting Network performance can be defined as estimating the future available bandwidth between grid sites across wide-area networks. To have a more efficient use of the resource, our job scheduling infrastructure and replica selection relies heavily on prediction of network performance. However, the complexity of gathering end-to-end network performance of resources would be increased with the number of grid sites, even if using some particular technology can reduce complexity away from $N^2$-N, such as in NWS that network sensors are organized hierarchically [8]. Because the wide-area links often are orders of magnitude slower than links of local networks, bandwidth

within cluster would be broader than WANs. To further reduce bandwidth consumption from probes for prediction, we propose a model to simplify current network performance prediction. It only measures bandwidth of WANs, shown as dotted lines in Figure 3.1.

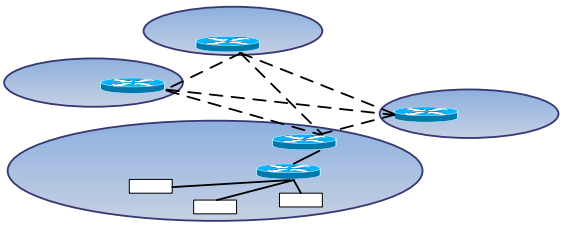


Figure 3.1: cluster-to-cluster Bandwidth

We assume each cluster has a leader or gateway to probe other cluster leaders. When communication is across two clusters, each pair of sites between two clusters has semantically related flows that share a common intermediary path, typically between first- and last-hop routers. Moreover, bottleneck bandwidth of a network path often occurs in wide-area links. With less accuracy, then, it is enough to probe a single clusterA-clusterB process pair to determine what the performance of any connection between clusterA and clusterB will be. In addition, while communication is within a cluster, each grid site observes approximately the same network performance, so the influence of bandwidth within a cluster on scheduling and replication can be ignored. This simple model that gathers a set of cluster-to-cluster performance would require $C^2$-C probes, where C is the number of clusters.

### 3.2. Hypothesis

Network performance can be converted into cost model. There are three factors that affect scheduling of job *j*: *S*, *R*, *Q*, where $S_j$ is the site to which the job *j* is scheduled, *R* is the list of LFN of replicas needed by the job and *Q* is the queuing latency for a job *j* at the site $S_j$. The replicas needed to execute this job are represented as $R = \{LFN_1, LFN_2, \ldots, LFN_n\}$. For a grid site $S_j$, we divide the replicas into three subsets according to the availability of $LFN_i$ in $S_j$. The first subset is on-site set $R^j_{on}$ that contains all the locally available replicas. The second subset is intra-site set $R^j_{intra}$ that contains the rest of replicas that can be found in the local cluster *C*. The third subset is inter-site set $R^j_{inter}$ that contains the other replicas that must be accessed from other clusters. For each $LFN_i$ in $R^j_{inter}$, assume the bandwidth from $S_j$ to $PFN_i$ (to the site that $LFN_i$ resides) is $B_{ji}$. Then the time needed to retrieve $PFN_i$ to $S_j$ is $|LFN_i|/B_{ji}$, where $|LFN_i|$ denotes the size of replica denoted by $LFN_i$. We define some cost terms.

Inter-cluster-communication-cost( $IrC^j_x$ ): If the job *j* is dispatched to cluster *x*, the cost of inter-communications would be calculated by using the cluster-to-cluster bandwidth.

$$IrC^j_x = \frac{1}{\alpha_j} \times \sum_{for\ all\ LFN_i\ in\ R^j_{inter}} \frac{|LFN_i|}{B_{ji}} \quad (1)$$

where $\alpha_j$ is a constant reflecting the degree of parallelism in $S_j$ for replica downloading. $IrC^j_x$ represents the time needed to have all the replicas in $R^j_{inter}$ available locally in $S_j$.

Intra-cluster-communication-cost( $IaC^j_{Sj}$ ):For job *j*, the cost of intra-communications at site $S_j$ is represented as the total file size of $R^j_{on}$, since bandwidth in the local cluster is assumed to be plentiful and roughly the same from sites to sites.

$$IaC^j_{Sj} = \sum_{for\ all\ LFN_i\ in\ R^j_{on}} |LFN_i| \quad (2)$$

We assume that each file has the same size, so $IaC^j_{Sj}$ can be regarded as $|R^j_{on}|$ which denotes the number of replicas in $R^j_{on}$.

**Queuing latency ($Q_j$):** If job *j* is going to be scheduled to $S_j$ in cluster *x*, queuing latency $Q_j$ will be the cost of running all the jobs that have queued at $S_j$. Therefore,

$$Q_j = \sum_{k=1}^{m} (IrC^k_x + IaC^k_x) \quad (3)$$

where 1,2,…,m are jobs queued before job j.

### 3.3. HCS (Hierarchical Cluster-based Scheduling) Algorithm

Previous job schedulers except Random scheduler search all resources to find the best one that has the lowest cost. HCS improves traditional schedulers in two aspects. First, HCS takes into account hierarchical cluster grid structure and all of data replicas owned by a cluster. Figure 3.2 is a simple example. There is a grid job that requires four files for execution and the four files are distributed over four clusters. If we schedule a job to the cluster based on highest hit ratio of required replicas (most of required replicas for the job available within cluster), like clusterB, the job execution time and the number of inter-cluster-communications would be reduced since access data can be faster (broader bandwidth within a cluster). In contrast, if scheduling a job to the cluster with few of required replicas, like clusterD, the number of inter-cluster-communications and access latency would be increased. It is possible that the amount of replica in one cluster is more than the other clusters but the total replicas size may be smaller, scheduling a job by using the number of replicas is inexact. Thus, to distribute the jobs to different sites, we propose to schedule jobs based on the cost model described in that last section.

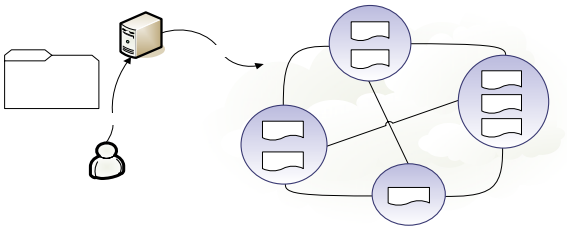

Figure 3.2: As example in Cluster Grid

Second, searching the best site from a huge amount of distributed sites would lead to long latency. HCS uses a hierarchical tree to schedule a job and minimize the overhead of searching for the suitable site, as shown in Figure 3.3. There are two-step decision processes. The first step selects a cluster to minimize the inter-cluster-communication-cost ( $IrC_C^j$ ). Referring back to the example shown in Figure 3.2, the values of $IrC_C^j$ for each cluster are:

(1) In clusterA, it needs to access File2 and File4. The best PFNs of File2 and File4 are both from clusterC that has minimum data latency ( $IrC_{ClusterA}^j$ = 4 sec).

(2) In ClusterB, it has most of the required replicas and only needs to access File3. But external bandwidth may be congested. Accessing the best PFN of File3 from clusterA, ( $IrC_{ClusterB}^j$ =5sec).

(3) In ClusterC, it lacks File1 and File3. The best PFNs of File1 and File3 are in clusterA ( $IrC_{ClusterC}^j$ =4sec).

(4) In ClusterD, it needs to access File1, File2, and File4 for job execution. The latency of moving File1 from clusterA, File2 and File4 form clusterC is $IrC_{CusterD}^j$ =7sec.

More than one cluster has minimum value of $IrC_C^j$ (clusterA and clusterC), in this situation, we will select one cluster randomly. Therefore, the job is scheduled onto clusterA or clusterC. This example shows that the cluster that has highest match of required replicas may not be the optimal solution.

After the suitable cluster is selected from Cluster Grid, the second step selects the best site Sj from local cluster based on the combined cost of moving replicas into the site Sj (intra-cluster-communication-cost) and the wait time in the queue in the site Sj (Queuing latency). The job is scheduled onto the site which has the minimum combined cost.

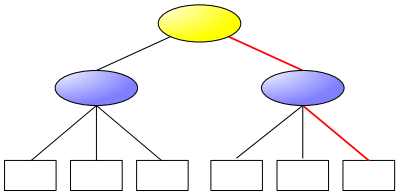

Figure 3.3: HCS Job Scheduling

### 3.4. HRS (Hierarchical Replication Strategy) Algorithm

After a job is scheduled to $S_j$, the requested data will be transferred to $S_j$ to become replicas. HRS (Hierarchical Replication Strategy) then determines how to handle this replica, as shown in Figure 3.4. If there is enough disk space, the replica is stored. Otherwise, if this replica is from a site in the local cluster, it is only stored in the temporary buffer and will be deleted after the job completes. If this replica is from other clusters, occupied space will be released to make room for this new replica, as presented by procedure 1 in Figure 3.4. The first choice to be removed is the replica that already exists in other sites in the same cluster. After all these locally available replicas are deleted, if the space is still insufficient, least frequently used replica will be the next target for removal, and so on until enough space is available. To conclude, HRS 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 not have to replicate them again later.

```
HRS replication strategy :
1. if (the needed replica is not in a site)
      if (replica is in the same cluster)
            select the replica with minimum intra-cluster-communication-cost within cluster.
      else
            select the replica with minimum intra-cluster-communication-cost between cluster.
2.    if (enough available space in local storage to store new replica)
            Store it;
      else {
3.          if (new replica is duplicated in other sites within region) {
                  Terminate optimizer; // avoid duplication in the same cluster
              }else {
                  for (each file in local storage) {
                    if (file is duplicated in other sites within region)
                              delete duplicated file;
                          if (enough free space to store new replica)
                              break;
                    }
              }
4.          if (!enough free space) {
                  using LFU replacement algorithm to delete unpopularity files
                  until has enough available space.
                  if (enough free space)
                        break;
            }
      }
      if (enough free space)
```

```
        Store new replica;
}
```

Figure 3.4: HRS replication strategy

### 3.5. HRS vs. BHR (Bandwidth Hierarchy based Replication)[16]

HRS replication strategy uses the concept of "network locality" as BHR [7]. The difference between HRS and BHR can be observed in two aspects. First, required replica within the same cluster is always the top priority used in HRS, 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 HRS will avoid inter-cluster-communications and be stable in hierarchical network architecture with variable bandwidth. Second, HRS considers popularity of replicas at site level, while BHR is based on region level.

## 4. Experiments

We use OptorSim to evaluate the performance of different combinations of job scheduling algorithms and replication strategies. OptorSim was developed to mimic the structure of a real Data Grid. All of general components are included into it with an emphasis on file access optimization and dynamic replication strategies. We have modified some components and embedded HCS and HRS modules in OptorSim to exactly match our needs. Behavior of OptorSim is set up and controlled by using configuration files. We describe in turn the simulation framework, experiments performed, and results.

### 4.1. Simulation Framework

To simplify the requirements, data replication approaches in Data Grid environments commonly assume that the data is read-only. It means that they can be replicated without having to assure change propagation back to the master copy. This is reasonable assumption as it is discussed in several scenarios [9] [10]. Consequently, all replicas are consistent. To prevent that all copies of a same file are deleted, for each file, there is one master file that contains the original copy of data samples and cannot be deleted by the replication strategies.

### 4.2. Experimental Environment

For the experiments, the cluster grid topology of the simulated platform is given in Figure 4.1 and this topology is from the simulation architecture of BHR. Node 35 holds all the master files at the beginning of the simulation. Each dotted line between two nodes shows the inter-cluster communication.

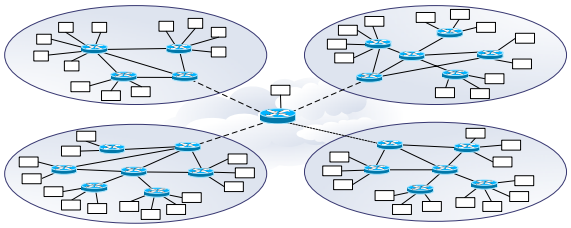

Figure 4.1: Topology of the simulated platform.

Table 1 specifies the simulation parameters used in our study. While running, jobs were randomly picked from 50 job types based on probability of each job, then submitted to the Resource Broker at regular intervals until 1000 jobs are submitted. Thus, some job types would occur frequently so that certain required replica access repeatedly.

Table 1: Simulation parameters

| Topology Parameter | Value |
|---|---|
| No. of cluster | 4 |
| No. of sites in each cluster | 13 |
| Storage space at each site | 50GB |
| Connectivity Bandwidth | 1000 Mbps (WAN)<br>1000 Mbps (LAN) |

| Grid Job Parameter | Value |
|---|---|
| No. of jobs | 1000 |
| No. of job types | 50 |
| No. of file accessed per job | 15 |
| Size of single file | 1GB |
| Total size of files | 750GB |

Files are accessed sequentially within a job without any access pattern. HCS will be compared with an OptorSim scheduler that searches all sites to find available CE by using a combination of access cost for the files and the queue length of waiting jobs, called QAC (Queue Access Cost). QAC performs better than other scheduler in OptorSim [11]. Additionally, HRS will be compared with LRU (Least Recently Used), LFU (Least Frequently Used), BHR (Bandwidth Hierarchy based Replication). The LRU algorithm always replicates and then deletes those files that have been used least recently. Similarly, LFU deletes the least frequently accessed file in recent past. We ran a total of six simulation experiments, which two kinds of scheduling policy combined four kinds of replication strategies. For experiment, we measure:

(1) Total job execution time (queuing time+ access latency+ executing time)

(2) Number of inter-communications

(3) Computing resource usage: the percentage of time that CEs are in active state at the period of job completion.

### 4.3. Experiment Results and Discussion

The following figures show the achieved results to complete 1000 jobs for each combination of the data replication and job scheduling algorithms. For replication strategies, LRU and LFU show similar performance in Figure 4.2. [1] shows the same results. We implement BHR replication strategy into OptorSim. Total job execution time is about 30% faster using BHR optimizer than LRU and LFU. Furthermore, we take benefit from network –level locality of BHR and simplify its replica replacement model. Thus, HRS successfully accelerates total time up to 40%.

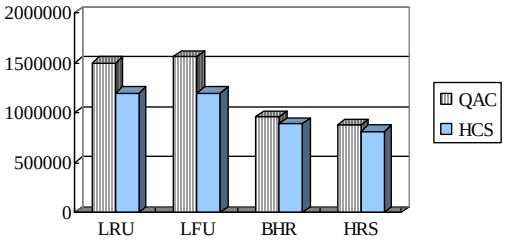


Figure 4.2: Total job execution times for various job scheduling and replication algorithms.

For job scheduling policy, HCS can improve performance about 10%~20%. The reason is that LRU and LFU do not take hierarchy network structure into account, so that HCS can reduce access latency and reduce 20% of total job execution time.

However, BHR and HRS both consider hierarchy of bandwidth in Internet, improved space of HCR is within limits. Total job execution time is about 10% faster using HCS job scheduling with BHR or HRS. HCS combined with HRS is about 50% less than QAC with LRU or LFU.

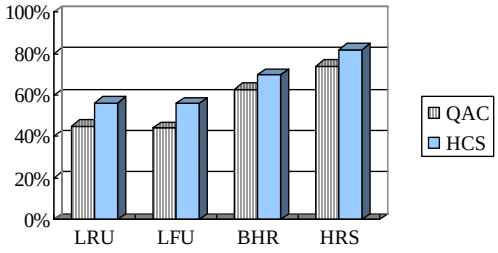


Figure 4.3: Computing resource usage.

As Figure 4.3 illustrates, computing resource usage is the percentage of time that CEs are in active state. It depends on job execution time, thus, computing resource usage shows almost the same performance in Figure 4.2. In the same simulation, total job execution time is decreased, computing resource usage is relative to increase.

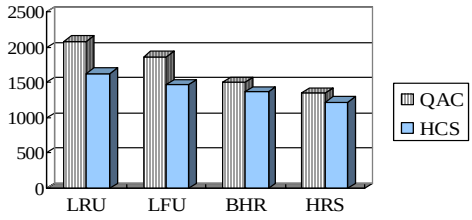


Figure 4.4: Number of inter-communications.

Based on Cluster Grid, HCS and HRS both reduce the number of inter-communications at about 20%, as shown in Figure 4.4. The results show that HCS and HRS save successfully on bandwidth over Internet.

### 4.4. Discussions

To analyze the distribution of jobs easily, there is a simplified example built in four clusters, each cluster has three grid sites and 500 jobs. HCS schedules jobs to certain specific sites and specific cluster. After all jobs are done, it can be observed that the distribution of jobs centralizes in some sites shown in Figure 4.6 (a). Similarly, the remainder of file on the site would belong to fixed file types.

On the contrary, the job distribution of QAC is out of order shown in Figure 4.6 (b). One site may have executed every job type. Files in that site are changeful.

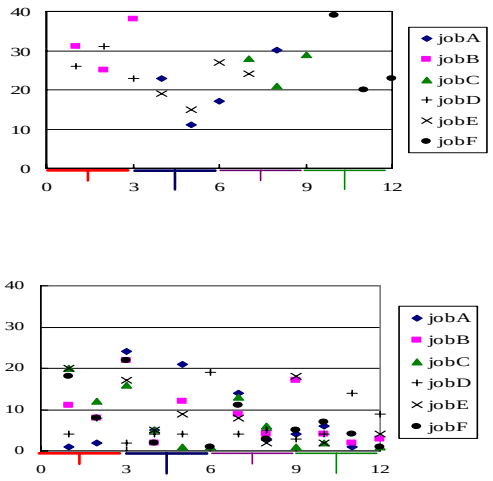


Figure 4.6: 500 jobs distribution (a) HCS with HRS (b) QAC with LFU.

HCS might lead some specific sites to heavy load, if a large amount of that certain job is submitted to grid. Also, some sites would be in starvation. However, scheduling jobs to site or cluster without data present spent more access latency than queue time at site with data and heavy load, since network bandwidth still fail to keep up with computing capacity, specifically the size of data is from terabyte to petabyte.

## 5. Conclusions and Future Work

We have addressed the problem of data movement operations in cluster grid environment. To achieve good network bandwidth utilization and data access time, we reduce the amount of inter-cluster communications. In support of this investigation, we propose a job scheduling policy (HCS) that considers not only computational capability and data location but also cluster information, and a dynamic replica optimization strategy (HRS) where the nearby data is the top priority to access then generating new replicas.

The simulation results show, first of all, that HCS and HRS both get better performance of inter-cluster bandwidth than other scheduling policy and replica strategies. Second, we can achieve particularly good performance with HCS in which jobs are always scheduled to cluster where most of data are located, and a separate HRS process at each site access dataset within the same cluster. Experiment data showed HCS scheduling with HRS replica strategy have an almost 50% reduction in total job execution time from the traditional algorithms such as LRU or LFU. Future work will implement our HCS job scheduling and HRS replication strategy into Taiwan UniGrid [12].

Acknowledgements: This research is supported in part by NSC under contract number 93-2213-E-259-013 and 93-2213-E-259-014. The authors would also

like to acknowledge the National Center for High-Performance Computing in providing resources under the national project “ Taiwan Knowledge Innovation National Grid”.

## 6. References

[1] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, “The Data Grid: Towards an Architecture for Distributed Management and Analysis of Large Scientific Datasets”, *Journal of Network and Computer Applications*, vol. 23, pages 187-200, 2000.

[2] W. H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, “Simulation of Dynamic Grid Replication Strategies in OptorSim”, *Proceedings of the Third ACM/IEEE International Workshop on Grid Computing (Grid2002)*, Baltimore, USA, vol. 2536 of *Lecture Notes in Computer Science,* pages 46-57, November 2002.

[3] I. Foster, and K. Ranganathan, “Identifying Dynamic Replication Strategies for High Performance Data Grids”, *Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing*, vol. 2242 of *Lecture Notes on Computer Science*, pages 75-86, Denver, USA, November 2002.

[4] I. Foster, and K. Ranganathan, “Decoupling Computation and Data Scheduling in Distributed Data-intensive Applications”, *Proceedings of the 11th IEEE International Symposium on High performance Distributed Computing (HPDC-11)*, IEEE, CS Press, pages 352-358, Edinburgh, U.K., July 2002.

[5] E. Deelman, H. Lamehamedi, B. Szymanski, and S. Zujun, “Data Replication Strategies in Grid Environments”, *Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing (ICA3PP'2002)*, IEEE Computer Science Press, pages 378-383, Bejing, China, October 2002.

[6] H.H. Mohamed and 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, pages 287-298, San Diego, California, USA , September 2004.

[7] Sang-Min Park, Jae-Hoon Kim, Young-Bae Go, and Won-Sik Yoon, "Dynamic Grid Replication strategy based on Internet Hierarchy", Lecture Note in Computer Science, *International Workshop on Grid and Cooperative Computing*, vol. 1001, pp.1324-1331, Dec. 2003.

[8] Jim Hayes, Neil T. Spring, and Rich Wolski, “The network weather service: a distributed resource performance forecasting service for metacomputing”, *Future Generation Computer Systems*, Vol.15 n.5-6, Pages 757-768, October 1999.

[9] Wolfgang Hoschek, Francisco Javier Jaén-Martínez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. "Data Management in an International Data Grid Project," *Proceedings of First IEEE/ACM International Workshop on Grid Computing (Grid'2000)*, Lecture Notes in Computer Science, Vol. 1971, pages 77-90, Bangalore, India, December 2000.

[10] P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, “Advanced Replica Management with Reptor”, *Proceedings of 5th International Conference on Parallel Processing and Applied Mathemetics (PPAM 2003)*, pages 848-855, Czestochowa, Poland, September 2003.

[11] DG Cameron, AP Millar, and C. Nicholson, “OptorSim: a Simulation Tool for Scheduling and Replica Optimisation in Data Grids”, *Proceedings of Computing in High Energy Physics*, CHEP 2004, Interlaken,

Switzerland, September 2004.

[12] Taiwan UniGrid Project Portal Site, 2003, http://unigrid.nchc.org.tw/