International Journal of Innovative Computing, Information and Control
Volume **8**, Number **9**, September **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **6271–6282**

GSWAP: A DATA EXCHANGING PARTITION FOR THE EXECUTION OF GRID JOBS

Liang Hu[1], Lin Lin[1], Xilong Che[1] and Changwu Li[2,*]

[1]Department of Computer Science and Technology
Jilin University
No. 2699, Qianjin Avenue, Changchun 130012, P. R. China
hul@jlu.edu.cn; linlinjlu@gmail.com; chexilong@jlu.edu.cn

[2]Dalian Polytechnic University
No. 1, Qinggongyuan, Dalian 116034, P. R. China
[*]Corresponding author: licw@dlpu.edu.cn

Received June 2011; revised November 2011

Abstract. *Grid aggregates heterogeneous resources over Internet to execute large-scale jobs. In the job execution, data is usually downloaded to the computing sites before the data processing. The batch mode of data transfer and processing might lead to the waiting of the reserved resources and the decrease of the system efficiency. Motivated by the online video service (in which the processing rate of the data matches the transfer rate), distributed file system is introduced in grid job execution to make the data transfer and data processing parallel. After analyzing and comparing the different types of distributed file systems, GSwap is proposed to deliver a data exchanging space for the job execution based on NAS. In addition, a new replication strategy – LTS strategy – is proposed and utilized in GSwap to overcome the drawbacks of NAS file sharing. With LTS, the performance and the availability of GSwap are improved. The evaluation shows that the performance of job execution has an obvious promotion with the data exchange under GSwap, and the performance has a further improvement with replication strategies.*

Keywords: Grid, Distributed file system, Performance optimization, Replication strategy

1. Introduction.

Grid aggregates heterogeneous resources over Internet to gain huge capacity of computation and storage [1]. It delivers dynamic resource sharing and co-operative problem solving among multiple Virtual Organizations. In recent years, Grid is widely used in scientific computation and industrial simulation as a high cost-effective solution. Grid job is a common method to execute applications in grid computing [2]. Grid midware provides open interfaces to invoke the resources over Grid. Thus, grid applications can be implemented by calling the interfaces as grid jobs.

As the data set involved in the jobs becomes larger and larger, the data access rate becomes the bottleneck of performance. Data involved in grid jobs is usually stored as files. In the process of file stage-in, the files would be downloaded to the executing sites. In this case, it makes the file transfer and the data process execute in batch mode. Thus, the reserved computing resources might be idle during the file transfer, and the efficiency of grid is brought down.

Motivated by online video services in which the data process rate matches (or is lower than) the data transfer rate, file sharing of distributed file system is introduced to promote the efficiency of job execution. In order to optimize the access utilizing file sharing, as well as manage the data involved and produced in the process of job execution, Grid Swap (GSwap) is proposed in this paper. It is composed by several grid sites and allows the sites to join and quit freely. GSwap provides the data exchanging space for Grid applications. It functions as the extension of local storage by mounting the storage spaces of remote sites that hold the files involved in the jobs, which is similar as Swap partition in Linux system functions as the extension of memory using local disk. Meanwhile, GSwap manages the storage resources shared by the grid sites and the files involved in the job execution. It maintains the availability of the files, the load balancing of the file servers (grid sites), and the joining and exiting of the sites.

The rest contents of this paper are organized as follows: (1) the related researches are introduced in Section 2; (2) the functions of GSwap are detailed and the architecture of GSwap is designed in Section 3; (3) the implementation of the prototype of GSwap is illustrated and the key modules are described in Section 4; (4) the comparison of the executing performance between GridFTP and GSwap is made in Section 5, and the performance in the different replication strategies under GSwap is compared, too; and (5) the conclusion is made and the future work is approved in Section 6.

2. **Related Work.**

The executing performance of grid jobs can be improved in several ways, and a lot of researches were made for this purpose.

Data transfer rate between grid sites can significantly affect the performance of job execution. GridFTP [3] is a high performance transfer protocol. It is advanced for high-bandwidth wide-area network based on FTP, supporting stripe mode [5] and the secure authentication using PKI [6]. It provides multi-stream data transfer in the network with limited bandwidth. At present, GridFTP acts as the basic transfer protocol in many kinds of grid midware. [4] proposes an intelligent file transfer protocol, in which the slide window algorithm is extended to realize the different data transfer mode and the FTP is elevated to a semantic web service level to simplify and automate data grid management.

Scheduling strategy of jobs in distributed system is an important factor relevant to the performance of system. The essence of job scheduling in grid is mapping the jobs to the resources. The factors such as estimated time of jobs and the state of resources are always considered in the scheduling strategies [7-9]. Data distribution in the grid is taken into consideration in the scheduling, and data-aware based strategy is researched [10]. Data-aware strategy dispatches jobs to the sites that have lower latency and higher bandwidth to the sites storing the needed data in the jobs. In this case, data transfer rate in the job execution is improved, and the efficiency of the job execution is promoted. Data transfer rate mainly depends on the network topology in the grid environment. In [11], a topology inference algorithm is given to serve as a key to build network-aware jobs.

Data replications improve the data access efficiency and balance the load in Grid [12-15]. In [12], six basic replication strategies and three file access patterns are introduced by Ranganathan and Foster. Such strategies are proposed on top of the file access frequency. They are compared by measuring average response time and the total bandwidth consumed under the three patterns. Many replication strategies are advanced based on their work. In [13], network bandwidth is considered in the file replication algorithm to improve the performance of replication strategy. In [14], a dynamic data replication mechanism – Latest Access Largest Weight (LALW) – is proposed. The overload of the network can be avoided in this way. [15] anticipates the file access frequency as the direction for replication strategies.

Grid file system [31,32] has the feature of durability, which means the data saved in it is durable. However, GSwap only provides data exchange space. Thus, the durability is not necessary in GSwap. It is enough for the files in it to be available during the applications related to them are running.

3. Architecture of GSwap.

GSwap delivers the space of data exchanging for the submission and execution of grid jobs. It is a temporary space rather than a file system, holding the files involved in the jobs. Actually, GSwap is organized as a "sub-grid" inside the grid.

GSwap allows users and jobs to access the data in the shared files directly using file sharing (rather than file downloading). Meanwhile users can also share storage resources to GSwap voluntarily. In this case, the files can be directly accessed by the other sites without the file stage-in, while the output of the grid jobs can be directly written to GSwap without the file stage-out. Figure 1 shows the principle of the data access in GSwap.

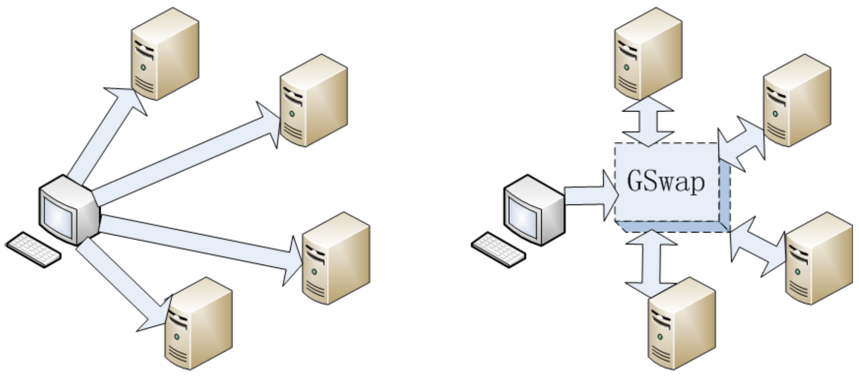


FIGURE 1. The principle of data access in GSwap

Hence, the following functions should be implemented in GSwap:

(1) **Distributed file sharing across VOs:** GSwap delivers file sharing on top of distributed file system. Grid is a distributed system that comprises several VOs. The file sharing should be available across VOs.
(2) **Global file management:** The information of the files in GSwap should be published by the information services so that users can query the state of them. The information should be updated in time. So a file monitoring module inside GSwap is inevitable.
(3) **File replication management:** File replication can enhance the concurrency of the GSwap access, balance the load of sites, and improve the availability of GSwap. The replications should be well managed.
(4) **The unified interfaces:** GSwap is designed for job submission. It should be compatible with the job submission tools. All the accesses to GSwap invoke the unified interface.
(5) **Dynamic management of the participated sites:** Grid sites are allowed to join and quit GSwap. The participated sites should be managed. Meanwhile, a fault tolerant mechanism should be made to prevent the access failure of key files.

According to the functions above, GSwap is composed by the following modules: file access protocol module, monitoring module, replication module, fault tolerance module and information service. The architecture of GSwap is described in Figure 2.

(1) **File access protocol module** provides the low level protocols for the file sharing and file transfer. All the operations in GSwap depend on this module. The other modules invoke the protocols to read, write, delete and transfer the files in GSwap.
(2) **Monitoring module** monitors the grid and GSwap. It comprises of three parts: file monitoring system, job monitoring system as well as resource monitoring system. File monitoring system watches the change of the file systems of GSwap sites; job monitoring system monitors the states of the submitted jobs; resource monitoring

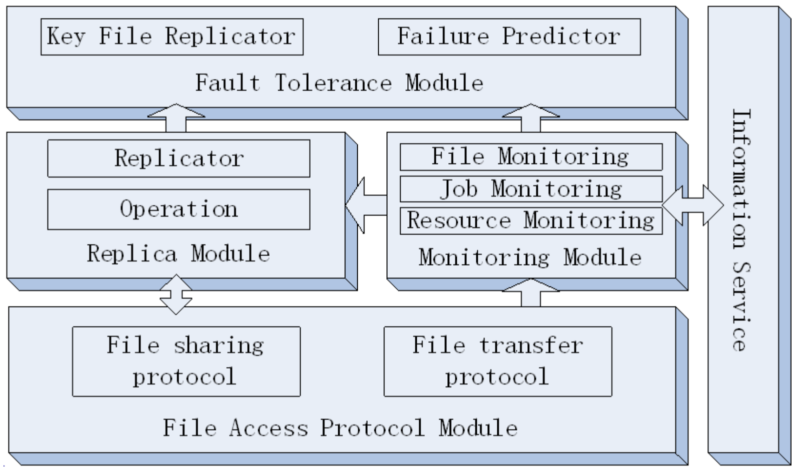


FIGURE 2. The relationship among the modules of GSwap

system records the system information of the grid, including the resource information of the sites and the network information.

(3) **Replica module** is in charge of the distribution of the file replicas in GSwap. It provides the operation interfaces to create, query, synchronize and delete the replicas. The replicator of this module applies replication strategies to deciding the distribution of replicas, according to the information gained by monitoring module.
(4) **Fault tolerance module** provides the strategies to prevent the failure of file access in GSwap. It invokes replica module to create replicas for the key files, in order to guarantee the availability of GSwap. It also predicts the failure with the monitoring information from monitoring module.
(5) **Information service** publishes all the information of the grid and GSwap, including the monitoring information and the replica information.

4. Distributed File Sharing in GSwap.

In this section, a prototype system of GSwap is introduced. It is designed and implemented on top of Globus Toolkit 5.0.1 (GT 5) [17,18].

4.1. Distributed file system.

Distributed file system allows access to files located on remote computers directly, just as they are located on local. Data access efficiency in the grid job execution might be promoted using distributed file sharing as file access protocol.

Distributed file system is developed from DAS (Direct Attached Storage) [19]. There are many types of distributed file system, including SAN (Storage Area Network) [20], NAS (Network Attached Storage) [21] and OBS (Object-base Storage) [22]. File sharing has different architectures in different types of distributed file systems, as is shown in Figure 3.

In SAN system, the file system index (logical view of file system) and the file operation management (physical view of file system) are put on a meta-data server. The storage devices are accessed by applications transparently on block level through high-speed network. As a result, the access rate in SAN is high. However, the security and the platform independence are hard to implement on block level. Additionally, as the amount of clients and storage devices increases, the overhead of the meta-data server rises rapidly. So it becomes the bottleneck of the system. In order to gain the flexible configuration to the file system, NAS emerged. In NAS system, the file system and storage are put into the remote server, and the server manages meta-data and the file operation for itself. The storage resources of the server are shared over the network with a file-based protocol. The

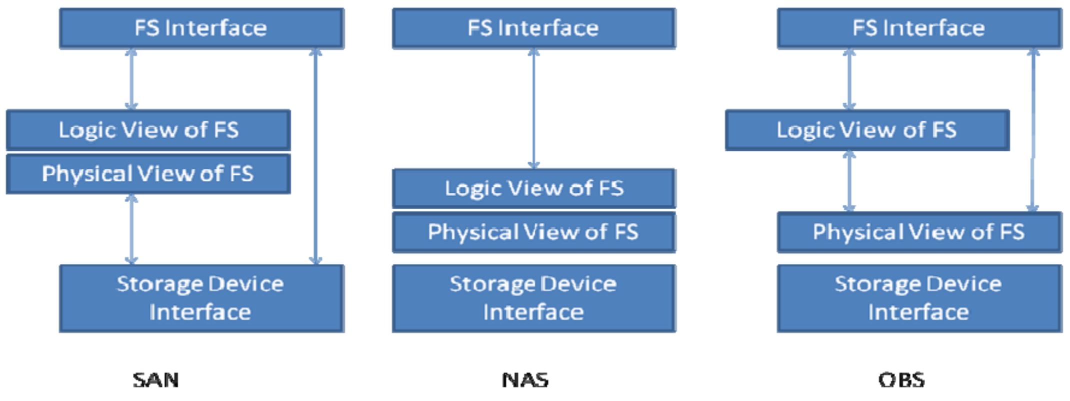


Figure 3. The comparison of SAN, NAS and OBS

security strategy and the platform independence are easy to implement on file level interface. However, as the data access path is too long, the overhead of the server is extremely high while it is accessed by several clients simultaneously. OBS is a compromise solution to data sharing between SAN and NAS. OBS uses object-based protocol. The meta-data server only manages the index of objects and maps the files to objects, while the other operations of storage management (including space allocation, free space management, etc.) are put into file servers. In this case, the performance of OBS is higher than NAS, and the security is easier to control than SAN. In addition, the overhead of the meta-data server and the file server declines, as well as the bottle-neck is eliminated.

However, not all the types of distributed file systems are suitable in GSwap. As a grid is composed of heterogeneous computers, the distributed file system utilized in grid should be platform independent to support all the computational resources of the grid, and the security of the file system is required. Therefore, SAN is not suitable in a grid environment despite the high access performance. Although OBS has advantages in both performance and security, and it can be platform independent, the data objects must be managed by the meta-server. Thus, it breaks the rules of autonomy of grid sites and non-centralized control in grid. So it is not an option for GSwap, neither. NAS provides the file sharing on top of local file system of the file server, which is easy to support various kinds of OS and control the security of file access. It performs the file sharing flexibly and satisfies all the requirements of Grid.

NAS has disadvantages, either. As the meta-data of the shared files are managed by the file server, the overhead of the server is high. The site might be overloaded when the simultaneous file access happens. Besides, the network latency restrains the performance of the file sharing. The maximum packet of NAS is 65535 bytes (in CIFS). If the access of a file requests more than 65535 bytes of data, the access is divided into several packets. In this case, the access rate is affected seriously by the network latency. Fortunately, these issues can be fixed by replicating the files among Grid sites. Replicating the files to different sites is equivalent to distributing the file access service to multiple sites [31]. Thus, the response time and the overhead of single file server decrease.

There are several kinds of NAS. NFS (Network File System) [23] and CIFS (Common Internet File System) [24] are typical NAS protocols. Considering that CIFS is easy to deploy on both Linux and Windows systems, the CIFS is used in GSwap. To simplify the implementation, security controlling is not considered in the prototype system. Every site of GSwap has a configuration file to configure the parameters as GSwap site. In the prototype system, only two parameters are configured:

gswap_location = *the path of the directory shared to GSwap*

gswap_size = *the maximum bytes can be allocated in the site*

The NAS server is started and the configuration file is parsed when the site joins in GSwap. According to gswap_location, NAS server shares the designated directory to GSwap. The shared space of the directory is limited by the minimum value of gswap_size and the size of idle local space.

4.2. **Monitoring system for GSwap.** Monitoring system comprises resource monitoring, job monitoring and file monitoring. All the information is gathered from grid sites except for the network information. Thus, the essence of the monitoring to grid is the aggregation of the monitoring information of every grid site, plus the network information. The information is aggregated by information service and database. As GSwap is a part of grid, the monitoring to GSwap is the aggregation of the monitoring information of GSwap sites. Figure 4 shows the architecture of the Monitoring system for GSwap.

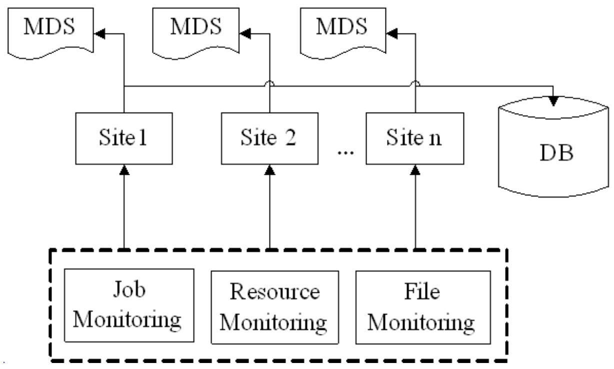


FIGURE 4. The architecture of the monitoring system for GSwap

4.2.1. *Resource monitoring module.* Resource monitoring module gathers the resource information, including computational resource information (CPU frequency, CPU occupancy, memory size, memory occupancy and average system load), storage resource information (total space, idle space and access rate) and network resource information (bandwidth and latency). Computational resource information and storage resource information are monitored by parsing the corresponding files in PROCFS. Network latency and bandwidth are measured by Iperf [25]. As all the resource information is monitored in PULL mode, heart-beat is used in this module which is encapsulated in a thread. Resource monitoring module works as Figure 5.

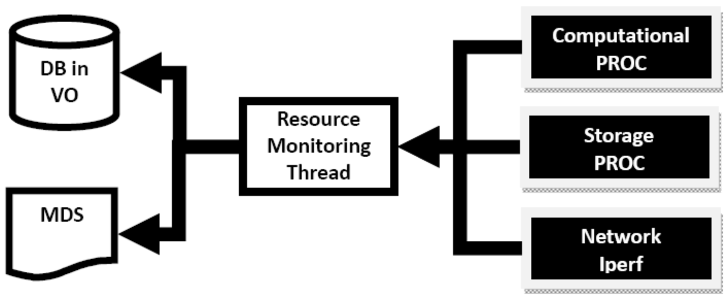


FIGURE 5. Resource monitoring module

4.2.2. *Job monitoring module.* Job monitoring module monitors the job submission and the information of job execution, including job information and process information. Job information describes the job, which comprises job ID (JID), submitting site, executing site and so on. Process information is gathered from the process of job execution, including the process ID (PID), the CPU and memory occupancy of the process and so on. In addition, the mapping from JID to PID is established to connect the two parts of information.

Jobs are submitted by GRAM 5 [26] in GT 5. The job information is gathered from the audit logging interface of GRAM 5. The audit logging interface delivers detailed information of job execution which covers most of the job information needed in job monitoring module.

After the job submission succeeds, the process to execute the job is established. A thread to monitor the process with PID is activated. The process information is gathered by parsing the files of the process in PROC. Figure 6 shows how the job monitoring module works.

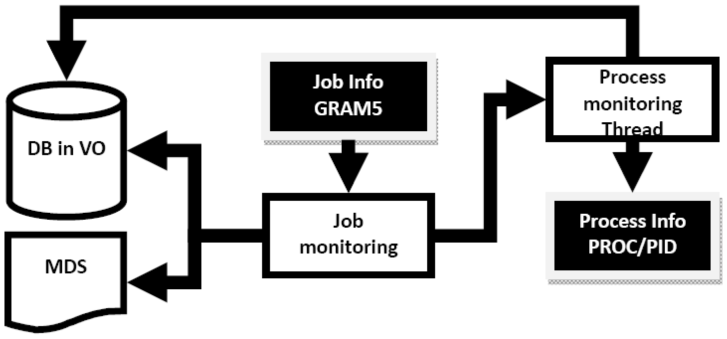


FIGURE 6. Job monitoring module

4.2.3. *File monitoring module.* File monitoring module comprises Local File Monitor (LFM) and Global File Monitor (GFM).

LFM watches the operation to the local files in **gswap_location** on the site, including file operation time, file operation type (create, move, delete, read, write, open and close). Inotify-java – a java version of Inotify [27] – is used to capture the operations. LFM counts the time of file operation with the notification from it. The hash codes of the files in **gswap_location** are updated when the files are modified. The file monitoring information is recorded in a local SQLite database, and the statistic information based on it, such as the access to files in a fixed time interval and the hash code of files, is published by MDS.

GFM provide the file replica information which is gathered from the statistic information of GSwap sites. GFM is built on top of Replica Location Service (RLS) [28].

4.3. **Replication strategy.** Replication Strategies are utilized in data grid to improve the performance and the availability of the system. However, data access in GSwap is different from that in data grid. The replication strategies used in GSwap should fit the features of GSwap:

(1) The length of the path from file server to accessing site: The access rate of NAS declines fast with the growth of network latency. If the length of the path from file server to access site is too long, the access rate of GSwap will be low. Thus, it should be ensured that the available file replica is not too far from the accessing site.

(2) Load of the file server: As the number of concurrent sites increases, the load of the NAS server grows violently. It will lead to the descending of the access rate. Thus, the distribution of file replicas should be considered according to the load of file servers.
(3) The consumption of network bandwidth: GSwap delivers concurrent data access in Grid. As GSwap is composed of several distributed sites of the Grid, the accessing sites and file servers share the network. So the creation of replica in GSwap might occupy the bandwidth for the file access. Thus, the unnecessary creation of the replica should be constrained in order to decline the consumption of the network bandwidth.
(4) Whether the site joins GSwap: GSwap allows sites to join and quit freely. Non-GSwap sites cannot take part in the file storage of GSwap. Thus, not all the grid sites can be used to create replica of file.

Data access in grid usually appears geographical and temporal locality [29]. In Ian foster's research [12], Cascading strategy shows the best performance in this pattern. It is chosen as a basic model in this paper. The features above are considered in Cascading strategy. Actually, the access rate provided in GSwap matches the data process rate of grid job in some kinds of applications. For example, the data process rate is so low in high ratio file compression that the access rate delivered in GSwap is able to satisfy the application. In this case, the execution time of the job is the same as the situation that the involved file is in local. Thus, even if the file replica is not in the accessing site, the performance is already optimum. As is shown in Figure 7, Site H, I and J are accessing sites. In Cascading, the file replicas are places on Site A, B, and D after two time intervals. If the replica strategy is continued according to Cascading model, the next replica would probably be created on Site G. However, if the data access rate already matches the data process rate under data sharing mode, it is unnecessary to continue the replica strategy. Nevertheless, Site D might be unable to burden the concurrent access by multiple sites, which would be the bottleneck of the system. In this case, the strategy should go on to create replica on Site G. Whether Site D is able to burden the concurrent access is decided by the load of Site D.

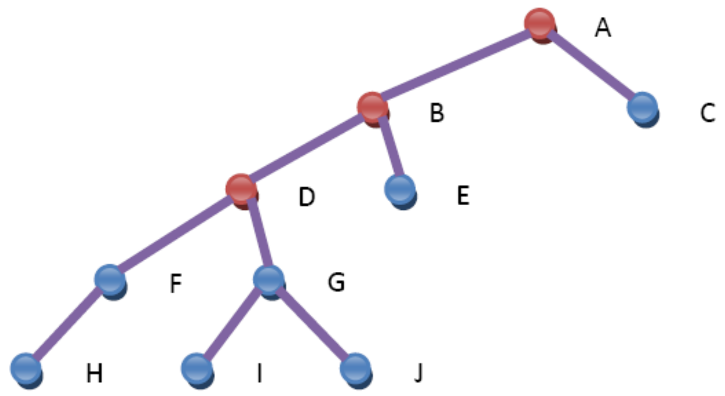


FIGURE 7. A middle state data distribution in cascading strategy

Based on the analysis above, Load and Topology Self-adaptive (LTS) strategy is proposed. For each site in GSwap, a replicator is deployed as a daemon. The replicator creates replicas according to the following rules:

Given k is an integer, T is a long integer, p is a real number and $0< p < 1$, h_1 and h_2 are real numbers and $0 < h_1 < h_2 < 3$, then,

(1) If a file in the site is accessed by its best client more than k times in time interval T and the promotion of access rate of the file is more than p in percentage to that of

the last replica (no replica mean the promotion is 100%), a file replication request is sent to the next GSwap site in the path to the best client.

(2) When a file replication request is received, if the load of the site does not exceed the threshold value h_1 and the site has more than 2 neighborhood sites, or if the site is the best client, the replica is created on the site. Else, the request is repeated to the next GSwap site in the path to the best client.

(3) If the load in the site exceeds the threshold value h_2, the replication request of the file that is most usually accessed in the current accessing files is sent to the next GSwap site on the path to the best client.

5. **Evaluation.** In this section, the performance of GSwap is evaluated by running a series of compute-intensive jobs under different modes in grid. The executing performance of grid job is compared under downloading mode and sharing mode, in order to verify the performance promotion under sharing mode. Then, the performance of GSwap is tested using different data replica strategies. The grid environment is composed by 8 Dell OptiPlex PCs with Ubuntu 9.10 and Globus Toolkit 5.0.1 and connected with 1000Mb Ethernet switch. The network latency is controlled by netem [30] to construct the tree topology of WAN. The topology of the grid is shown in Figure 8. Each edge between two sites means that 10ms is added to the latency.

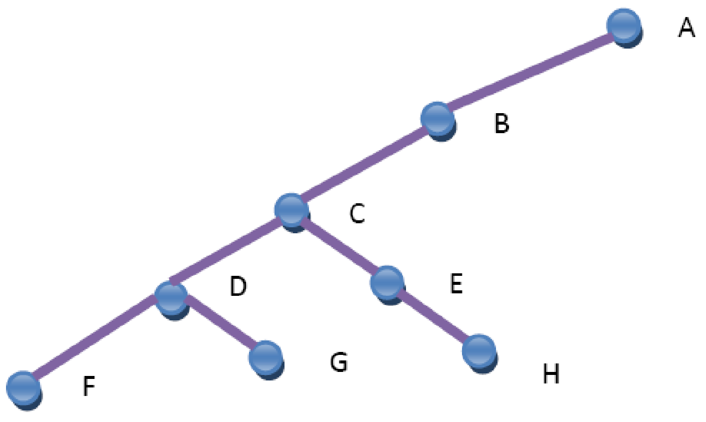
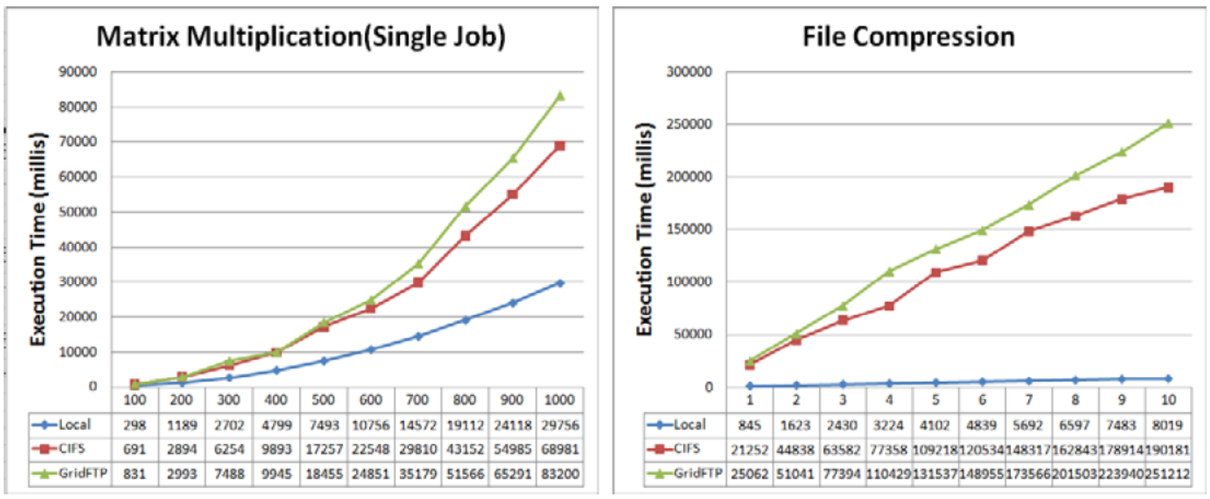


FIGURE 8. Topology of grid in the tests

Matrix multiplication and data compression are chosen as the compute-intensive jobs for the evaluation. Matrix multiplication is a typical kind of calculation in scientific problems. The scale of the calculation grows exponentially with the order of the matrixes. Data compression is another kind of data intensive computation in which the scale of computation grows with the size of the file linearly, and the file operations happen more frequently.

In this section, the order of the matrixes n grows from 100 to 1000 by 100. The contents of the matrixes are integers generated automatically from 0 to n^2. The files in the file compression have the same size of 13.7MB, and the amount of the files grows from 1 to 10 in the jobs.

5.1. **The performance comparison under two modes.** In this section, a series of matrix multiplication jobs and a series of data compression jobs are submitted to remote sites under either downloading mode (GridFTP) or sharing mode (NAS). The situation that data access in local disks is taken into consideration as a reference to the two modes. The environment of the test in this part is ideal: the network latency is less than 1ms, and the file server has maximum one client at same time.



	100	200	300	400	500	600	700	800	900	1000
Local	298	1189	2702	4799	7493	10756	14572	19112	24118	29756
CIFS	691	2894	6254	9893	17257	22548	29810	43152	54985	68981
GridFTP	831	2993	7488	9945	18455	24851	35179	51566	65291	83200

	1	2	3	4	5	6	7	8	9	10
Local	845	1623	2430	3224	4102	4839	5692	6597	7483	8019
CIFS	21252	44838	63582	77358	109218	120534	148317	162843	178914	190181
GridFTP	25062	51041	77394	110429	131537	148955	173566	201503	223940	251212

Figure 9. Comparison of the executing performance under different modes

Figure 9 shows the comparison of the executing performance under different modes. As is shown in Figure 9, the execution time of the jobs under sharing mode is more than that in local and less than that in downloading mode. It verifies the performance promotion of GSwap in an ideal condition.

5.2. **The performance test of GSwap.** This section tests the performance of GSwap with different replica strategies including Cascading strategy and LTS strategy.

The jobs used in this section comprise the two types of jobs in Section 5.1. The matrix multiplication jobs are divided into two parallel jobs and are submitted to Site F and Site G, and the file compression jobs are submitted to Site F and Site G randomly.

The size of the matrixes in matrix multiplication are all 1000×1000, and the size of the file set in file compression is 13.7MB. All the files have no replica and are saved in Site before the test. In the strategies, the threshold value k for best client is set to 2, and the time interval T is set to 180000ms. In LTS strategy, the load threshold h_1 is set to 1.5 and h_2 is set to 1.8. Figure 10 shows the performance comparison in matrix multiplication and that in file compression separately.

Figure 10 shows the performance promotion by LTS strategy. The average job execution time is obviously shorter in LTS strategy. And it can be inferred that the diffusing speed of the data in LTS strategy is faster than that in Cascading strategy. Therefore, LTS strategy is more efficient than Cascading strategy in GSwap.

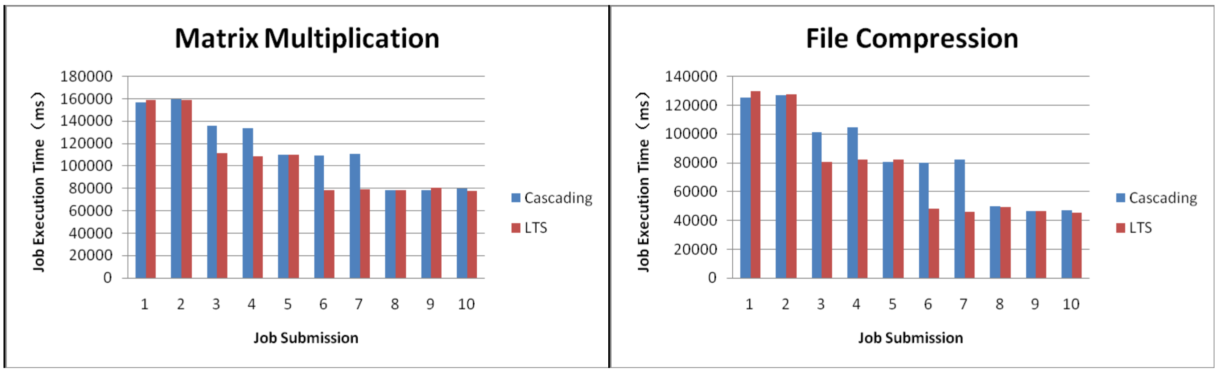


Figure 10. Performance of cascading and LTS

6. **Conclusions.** In this paper, distributed file system is applied in the remote data access of grid jobs, in order to promote the executing performance of grid jobs and improve the efficiency of grid system. After analyzing and comparing, NAS is chosen out of many types of distributed file system as the most suitable one for the data sharing in grid job. Then, GSwap is proposed based on NAS as the data exchanging space for grid jobs. It is composed by the sites in grid. In order to promote the availability and performance of GSwap, LTS strategy is proposed for data replication. After a series of tests, the promotion of executing performance of grid jobs under GSwap data sharing is verified. However, LTS is a simple strategy, and the performance of GSwap might have a further promotion after a deep research to the replication strategy. As LTS replicates the files according to the monitoring information, the decision of replication lags behind the job execution. By predicting the location of clients, the replication might be distributed to appropriate sites before the job execution, which is similar to the pre-reading mechanism of cache. On top of the prototype system, the issues on security control and the replica synchronizing should be considered in the future researches, too.

Acknowledgement. The work in this paper is supported by the National Natural Science Foundation of China (No. 60873235 and No. 61073009) and by Major Project of Chinese National Programs for Fundamental Research and Development (973 Program) (No. 2009CB320706) and National High Technology Research and Development Program of China (863 Program) (No. 2011AA010101).

REFERENCES

[1] I. Foster, Y. Zhao, I. Raicu and S. Lu, Cloud computing and grid computing 360-degree compared, *Grid Computing Environments Workshop*, pp.1-10, 2008.

[2] A. Suciu and R. Potolea, A taxonomy for grid applications, *International Conference on Automation, Quality and Testing, Robotics*, vol.3, pp.365-368, 2008.

[3] J. Bresnahan, M. Link, G. Khanna, Z. Imani, R. Kettimuthu and I. Foster, Globus GridFTP: What's new in 2007, *The 1st International Conference on Networks for Grid Applications, ICST*, Brussels, Belgium, pp.1-5, 2007.

[4] J. Wang and L. Huang, Intelligent file transfer protocol for grid environment, *Current Trends in High Performance Computing and Its Applications*, pp.469-476, 2005.

[5] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu and I. Foster, The globus striped GridFTP framework and server, *Conference on High Performance Networking and Computing*, Washington, DC, USA, pp.54-64, 2005.

[6] H. Li and Y. Wang, Public-key infrastructure, *Payment Technologies for E-Commerce*, pp.39-70, 2003.

[7] A. Moallem and S. A. Ludwig, Using artificial life techniques for distributed grid job scheduling, *Symposium on Applied Computing*, New York, NY, USA, pp.1091-1097, 2009.

[8] L. He, S. A. Jarvis, D. P. Spooner, X. Chen and G. R. Nudd, Dynamic scheduling of parallel jobs with QoS demands in Multiclusters and grids, *The 5th International Workshop on Grid Computing (GRID'04)*, Washington, DC, USA, pp.402-409, 2004.

[9] B. Duran and F. Xhafa, The effects of two replacement strategies on a genetic algorithm for scheduling jobs on computational grids, *Symposium on Applied Computing*, New York, NY, USA, pp.960-961, 2006.

[10] T. Kosar and M. Balman, A new paradigm: Data-aware scheduling in grid computing, *Future Generation Computer Systems*, vol.25, no.4, pp.406-413, 2009.

[11] T. Shirai, H. Saito and K. Taura, A fast topology inference: A building block for network-aware parallel processing, *The 16th International Symposium on High Performance Distributed Computing Monterey*, CA, USA, pp.11-22, 2007.

[12] K. Ranganathan and I. Foster, Design and evaluation of dynamic replication strategies for a high-performance data grid, *International Conference on Computing in High Energy and Nuclear Physics*, 2001.

[13] H. Sato, S. Matsuoka, T. Endo and N. Maruyama, Access-pattern and bandwidth aware file replication algorithm in a grid environment, *The 9th International Conference on Grid Computing, International Conference on Grid Computing*, Washington, DC, USA, pp.250-257, 2008.

[14] R. Chang, H. Chang and Y. Wang, A dynamic weighted data replication strategy in data grids, *International Conference on Computer Systems and Applications*, pp.414-442, 2008.

[15] M. Lei, S. V. Vrbsky and X. Hong, An on-line replication strategy to increase availability in data grids, *Future Gener. Comput. Syst.*, vol.24, no.2, pp.85-98, 2008.

[16] C.-T. Yang, C.-P. Fu, C.-J. Huang and C.-H. Hsu, FRCS: A file replication and consistency service in data grids, *International Conference on Multimedia and Ubiquitous Engineering*, pp.444-447, 2008.

[17] I. Foster, Globus toolkit version 4: Software for service-oriented systems, *IFIP Int. Conf. on Network and Parallel Computing, LNCS*, vol.3779, pp.2-13, 2006.

[18] *Globus Toolkit*, http://www.globus.org/toolkit/about.html.

[19] D. Sacks, Demystifying DAS, SAN, NAS, NAS gateways, fibre channel and iSCSI, *IBM Storage Consultant*, pp.6-19, 2001.

[20] T. Clark, *Designing Storage Area Networks: A Practical Reference for Implementing Storage Area Networks*, 2 Edition, Addison-Wesley Longman Publishing Co., Inc, 2003.

[21] Y. Deng and F. Wang, Exploring the performance impact of stripe size on network attached storage systems, *J. Syst. Archit.*, vol.54, no.8, pp.787-796, 2008.

[22] D. Nagle, M. E. Factor and S. Iren, The ANSI T10 object-based storage standard and current implementations, *IBM Journal of Research and Development*, vol.52, no.4, 2008.

[23] B. Pawlowski, S. Shepler, C. Beame, B. Callaghan, M. Eisler, D. Noveck, D. Robinson and R. Thurlow, The NFS version 4 protocol, *The 2nd International System Administration and Networking Conference*, Maastricht, The Netherlands, pp.94-113, 2000.

[24] C. Hertel, *Implementing Cifs: The Common Internet File System*, Prentice Hall Professional Technical Reference, 2003.

[25] E. Yildirim, I. H. Suslu and T. Kosar, Which network measurement tool is right for you? A multidimensional comparison study, *International Conference on Grid Computing (GRID '08)*, Washington, DC, USA, pp.266-275, 2008.

[26] *GT 5.2.1 GRAM5*, http://www.globus.org/toolkit/docs/5.2/5.2.1/gram5/#gram5.

[27] R. Love, Kernel korner: Intro to inotify, *Linux J.*, vol.139, 2005.

[28] A. L. Chervenak, R. Schuler, M. Ripeanu, M. A. Amer, S. Bharathi, I. Foster, A. Iamnitchi and C. Kesselman, The globus replica location service: Design and experience, *IEEE Trans. Parallel Distrib. Syst.*, vol.20, no.9, pp.1260-1272, 2009.

[29] J. Zhang and P. Honeyman, A replicated file system for grid computing, *Concurrency and Computation: Practice and Experience, Special Issue: Middleware for Grid Computing: Future Trends*, vol.20, no.9, pp.1113-1130, 2008.

[30] S. Hemminger, Netem – Emulating real networks in the lab, *LCA2005*, Canberra, Australia, 2005.

[31] L. Lindbäck, V. Vlassov, S. Mokarizadeh and G. Violino, Churn tolerant virtual organization file system for grids, *International conference on Parallel Processing and Applied Mathematics: Part II*, pp.194-203, 2009.

[32] F. Garcia-Carballeira, J. Carretero, A. Calderon, J. D. Garcia and L. M. Sanchez, A global and parallel file system for grids, *Future Gener. Comput. Syst.*, vol.23, no.1, pp.116-122, 2007.