A Dynamic Adjustment Strategy for File Transformation in Data Grids*

Chao-Tung Yang†, Shih-Yu Wang, and Chun-Pin Fu

High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University, Taichung, 40704, Taiwan
email: ctyang@thu.edu.tw

Abstract. In this paper, we propose a dynamic file transfer scheme with co-allocation architecture, called Dynamic Adjustment Strategy, a dynamic file transfer scheme with co-allocation architecture that reduce the file transfer times and improves the performance in Data Grid environments. Our approach reduces the idle time faster servers spend waiting for the slowest server, and decreases file transfer completion time. We also present a new toolkit, called Cyber-Transformer, with a friendly graphical user interface interface running on the client side that makes it easy for inexperienced users to manage replicas and download the files in Data Grid environments. We also provide an effective scheme for reducing the cost of reassembling data blocks.

1 Introduction

The term "Data Grid" traditionally represents the network of distributed storage resources from archival systems to caches and databases, which are linked using a logical name space to create global, persistent identifiers and provide uniform access mechanisms [4]. Data Grids aggregate distributed resources to resolve large-size dataset management problems [1, 2, 3, 5, 6, 7, 8, 9, 29, 30]. Increasingly, large collections of measured and computed data are emerging as important resources in many data-intensive applications.

Certain data-intensive scientific applications entail huge amounts of data that require data file management systems to replicate files and manage data transfers and distributed data access. Data grid infrastructure integrates data storage devices and data management services in grid environments consisting of scattered computing and storage resources, perhaps located in different countries/regions yet accessible to users [12].

Replicating popular content in distributed servers is widely used in practice [14, 17, 21, 30]. Recently, large-scale, data-sharing scientific communities such as those described in [1, 5] used this technology to replicate their large datasets over several

* This work is supported in part by the National Science Council, Taiwan R.O.C., under grants no. NSC95-2221-E-029-004 and NSC95-2218-E-007-025.

† Corresponding author.

sites. Downloading large datasets from several replica locations may result in varied performance rates. Bandwidth quality is the most important factor affecting transfers between clients and servers since download speeds are limited by the bandwidth traffic congestion in the links connecting the servers to the clients.

One way to improve download speed is to use replica selection techniques to determine the best replica locations [21]. This method selects the servers most likely to provide optimum transfer rates because bandwidth quality can vary unpredictably due to the sharing nature of the Internet. Another way is to use co-allocation technology [17, 21, 23, 24, 25, 26, 27, 28, 30] to download data. Co-allocation of data transfers enables the clients to download data from multiple locations by establishing multiple connections in parallel. This can improve the performance over single-server downloads and alleviate the Internet congestion problem [17]. Several co-allocation strategies were presented in our work [17]. An idle-time drawback remains since faster servers must wait for the slowest server to deliver its final block. Thus, reducing the differences in finish times among replica servers is important.

In this paper, we propose a dynamic file-transfer scheme with co-allocation architecture, called the Dynamic Adjustment Strategy, which reduces file-transfer times and also improves data transfer performance in Data Grid environments. Our approach can reduce file server idle times and decrease file-transfer completion times. We also present a new toolkit, called Cyber-Transformer, with a friendly client-side GUI interface integrated with the Information Service, Replica Location Service, and Data Transfer Service [25]. And we provide an effective scheme for reducing the cost of reassembling data blocks. Experimental results show that our approach is superior to previous methods and achieves the best overall performance. We also discuss combination cost and provide an effective improvement.

2 Background Review

2.1 Data Grid and Grid Middleware

In Data Grid environments, access to distributed data is typically as important as access to distributed computational resources [1, 2, 3, 4, 5, 6, 30]. Distributed scientific and engineering applications require transfers of large amounts of data between storage systems, and access to large amounts of data generated by many geographically distributed applications and users for analysis and visualization, among others.

The Globus Project [9, 11, 16] provides software tools collectively called The Globus Toolkit that make it easier to build computational Grids and Grid-based applications. Many organizations use the Globus Toolkit to support their applications. The composition of the Globus Toolkit can be pictured as three pillars: Resource Management, Information Services, and Data Management. Each pillar represents a primary component of the Globus Toolkit and makes use of a common foundation of security. GRAM implements a resource management protocol, MDS implements an

information services protocol, and GridFTP implements a data transfer protocol. They all use the GSI security protocol at the connection layer [10, 11, 13, 16].

2.2 Replica Management and Selection

Replica management involves creating or removing replicas in data grid sites [21]. A replica manager typically maintains a replica catalog containing replica site addresses and file instances. The replica management service is responsible for managing the replication of complete and partial copies of datasets, defined as collections of files.

Data Grid may contain multiple replica catalogs. The purpose of the replica catalog is to provide mappings between logical names for files or collections and one or more copies of objects in physical storage systems. The catalog registers three types of entries: logical collections, locations and logical files. Despite the benefits of registering and manipulating collections of files using logical collection and location objects, there may be a need for users and applications to characterize individual files. For this purpose, the Replica Catalog includes optional entries that describe individual logical files. Logical files are entities with globally unique names and one or more physical instances. The Catalog may optionally contain one logical file entry in the Replica Catalog for each logical file in a collection.

Replica selection [16] is used to select replicas from among the sites in a Data Grid [21]. The selection criteria depend on application characteristics. This mechanism enables users to efficiently manage replicas of data sets at their sites. The replica selection process commonly consists of three steps: data preparation, preprocessing and prediction. Applications then select replicas according to their specific attributes.

3 The Dynamic Adjustment Strategy

3.1 The Co-Allocation Architecture

Candidate replica locations are passed to the replica selection service [21], which was presented in a previous work [23, 24, 25]. This replica selection service provides estimates of candidate transfer performance based on a cost model and chooses appropriate amounts to request from the better locations. The architecture proposed in [17] consists of three main components: an information service, a broker/co-allocator, and local storage systems. The co-allocation architecture [7] is shown in Figure 1, which is an extension of the basic template for resource management. Applications specify the characteristics of desired data and pass the attribute description to a broker. The broker queries available resources and gets replica locations from an information service [6] and a replica management service [21] creates a list of the desired files physical locations. The co-allocation agent then downloads the data in parallel from the selected servers.

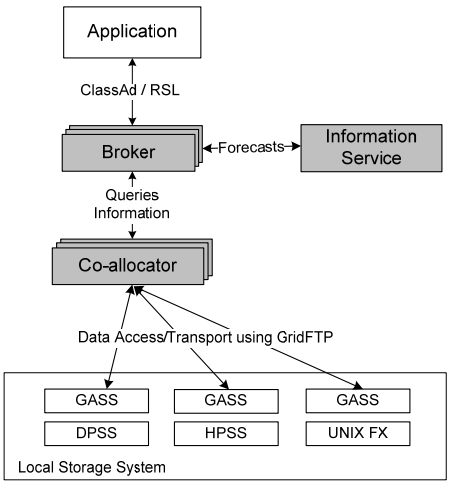

Fig. 1. The Co-Allocation Architecture in Data Grids

Data grids consist of scattered computing and storage resources located in different countries/regions yet accessible to users [8]. We used the grid middleware Globus Toolkit [16] as our data grid infrastructure. The Globus Toolkit provides solutions for such considerations as security, resource management, data management, and information services. One of its primary components, MDS [6, 11, 16, 26], is designed to provide a standard mechanism for discovering and publishing resource status and configuration information. It provides a uniform and flexible interface for data collected by lower-level information providers in two modes: static (e.g., OS, CPU types, and system architectures) and dynamic data (e.g., disk availability, memory availability, and loading) [15, 22]. And it uses GridFTP [1, 11, 16] to provide efficient management and transfer data in a wide-area, distributed-resource environment. We use GridFTP [1, 11, 16] to enable parallel data transfers.

As datasets are replicated within Grid environments for reliability and performance, clients require the abilities to discover existing data replicas, and create and register new replicas. A Replica Location Service (RLS) [4] provides a mechanism for discovering and registering existing replicas. Several prediction metrics have been developed to help replica selection. For instance, Vazhkudai and Schopf [18, 19, 20, 21] used past data transfer histories to estimate current data transfer throughputs.

In our previous work [23, 24], we proposed a replica selection cost model and a replica selection service to perform replica selection. In [17], the author proposes a co-allocation architecture for co-allocating Grid data transfers across multiple connections by exploiting the partial-copy feature of GridFTP. It also provides Brute-Force, History-Based, and Dynamic Load Balancing for allocating data blocks.

- Brute-Force Co-Allocation: Brute-Force Co-Allocation works by dividing files equally among available flows. It does not address bandwidth differences among the various client-server links.

- History-based Co-Allocation: The History-based Co-Allocation scheme keeps block sizes per flow proportional to transfer rates predicted by previous results of file transfer results.
- Conservative Load Balancing: One of their dynamic co-allocation is Conservative Load Balancing. The Conservative Load Balancing dynamic co-allocation strategy divides requested datasets into "*k*" disjoint blocks of equal size. Available servers are assigned single blocks to deliver in parallel. When a server finishes delivering a block, another is requested, and so on, until the entire file is downloaded. The loadings on the co-allocated flows are automatically adjusted because the faster servers will deliver more quickly providing larger portions of the file.

These co-allocation strategies do not address the shortcoming of faster servers having to wait for the slowest server to deliver its final block. In most cases, this wastes much time and decreases overall performance. Thus, we propose an efficient approach, called the Dynamic Adjustment Strategy, and based on the co-allocation architecture. It improves dynamic co-allocation and reduces waiting time, thus improving overall transfer performance.

3.2 The Dynamic Adjustment Strategy

Dynamic co-allocation is the most efficient approach to reducing the influence of network variations between clients and servers. However, the idle time of faster servers waiting for the slowest server to deliver its last block is still a major factor affecting overall efficiency, which Conservative Load Balancing and Aggressive Load Balancing [17] cannot effectively avoid. The approach proposed in the present paper, a dynamic allocation mechanism, called Dynamic Adjustment Strategy, can overcome this, and thus, improve data transfer performance.

Co-allocation technology [17] enables the clients to download data from multiple locations by establishing multiple connections in parallel. In our previous work [23], we proposed a replica selection cost model and a replica selection service to perform replica selection. We now propose a new data transfer strategy based on this model. It consists of three phases: initial phase, steady phase, and completion phase.

- Initial phase: We assign equal block sizes to all GridFTP servers. In this phase, our system determines the next block size for each replica server.
- Steady phase: As job transfers are completed, servers are assigned their next jobs. Jobs sizes are determined by multiplying the client bandwidth by the weighting.
- Completion phase: To avoid the generating excessively small job sizes, we set an end condition such that if the remaining target file size is smaller than the initial block size, it is transferred immediately.

To determine the initial block size, we set an upper bound that is dependent on the relation between the client's maximum bandwidth and the number of replica sources. Though multiple replicas can be downloaded in parallel, the gathered portions of files from different links must be transferred to the client in a single link. It is clear that the client's bandwidth could be bottleneck in co-allocation architecture. The formula for upper bound is:

$$initialPT \leq ClientMaxBandwidth / Number\ of\ Replica\ Source \quad (1)$$

In our previous work [23, 25, 26, 27, 28], we proposed a replica selection cost model in which we defined a formula for calculating the weighting. First, we get a score based on the states of the various server devices:

$$Score_i = P_i^{CPU} \times R^{CPU} + P_i^{Mem} \times R^{Mem} + P_i^{BW} \times R^{BW}, \text{ and } R^{CPU} + R^{Mem} + R^{BW}=1 \quad (2)$$

The parameters are:

- $Score_i$: the score for server i such that $1 \leqq i \leqq n$.
- P_i^{CPU}: percentage of server i CPU idle states [15]
- R^{CPU}: CPU load ratio defined by the user
- P_i^{Mem}: percentage of server i memory free space [15]
- R^{Mem}: memory free space ratio defined by the user
- P_i^{BW}: percentage of bandwidth available from server i to client (user node); current bandwidth divided by highest theoretical bandwidth [22, 24]
- R^{BW}: network bandwidth ratio defined by users.

After getting the scores for all server nodes, the system calculates the $weighting_i$:

$$weighting_i = Score_i / \sum_{k=1}^{n} Score_k \quad (3)$$

The weighting is then used to determine the size of the next job:

$$newPT_i = ClientBandwidth \times weighting_i \quad (4)$$

Where $newPT_i$ denotes the next job size for server i, and *ClientBandwidth* denotes the current client bandwidth.

When server i finishes transferring of a block, it gets a new job whose size is calculated according to the real-time status of server i. Each time, our strategy dynamically adjusts a job size according to source device loading and bandwidth. The lighter the loading a source device has, the larger job size it is assigned. We show experimental results and analyses that confirm our strategy in the next section.

4 Experimental Results and Analysis

In this section, we discuss the performance of our Dynamic Adjustment Co-Allocation strategy in a real data grid. We evaluate four co-allocation schemes: (1) Brute-Force (Brute), (2) History-based (History), (3) Conservative Load Balancing (Conservative) and (4) Dynamic Adjustment Strategy (DAS). We analyze the performance of each scheme by comparing their transfer finish times, and the total idle time faster servers spent waiting for the slowest servers to finish delivering the last block. We also analyze overall performances in the various cases.

We performed wide-area data transfer experiments using our GridFTP GUI client tool. We executed our co-allocation client tool on our testbed at Tunghai University (THU), Taichung City, Taiwan, and fetched files from four selected replica servers: one at Providence University (PU), one at Li-Zen High School (LZ), and the other one at Hsiuping Institute of Technology School (HIT). All these institutions are in Taichung, Taiwan, and each is at least 10 Km from THU. Figure 2 shows our Data Grid testbed.

In the following experiments, we set R^{CPU}, R^{MEM}, and R^{BW} in the ratio 0.1:0.1:0.8. We experimented with file sizes of 10MB, 50MB, 100MB, 500MB, 1000MB,

1500MB, and 2000MB. For comparison, we measured the performance of Conservative Load Balancing on each size using the same block numbers.

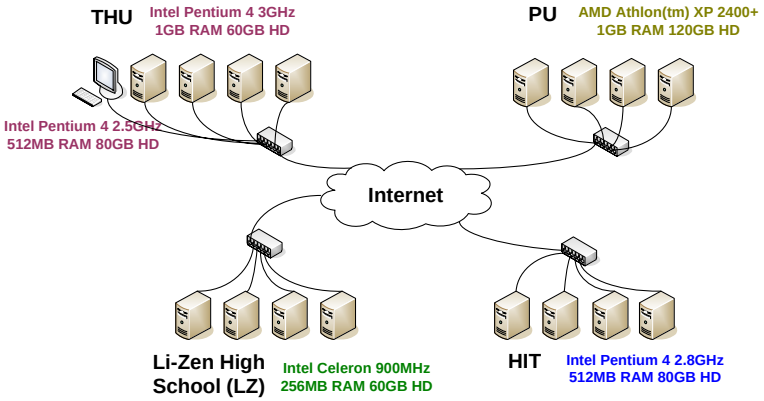

Fig. 2. Our Data Grid testbed

Table 1 shows average transmission rates between THU and each replica server. These numbers were obtained by transferring files of 100MB, 500MB, 1000MB, and 2000MB from a single replica server using our GridFTP client tool, and each number is an average over several runs.

Table 1. GridFTP end-to-end transmission rates from THU to various servers

Replica Server	Average Transmission Rate
HIT	61.5 Mbits
LZ	49.5 Mbits
PU	26.7 Mbits

We examined the effect of faster servers waiting for the slowest server to deliver the last block for each scheme. Figure 3 shows total idle times for various file sizes. Note that our Dynamic Adjustment Strategy performed significantly better than the other schemes on every file size. These results demonstrate that our approach efficiently reduces the differences in servers finish times.

Figure 4 shows total completion times in a detailed cost-structure view. Servers were at PU, LZ, and HIT, with the client at THU. The first three bars for each file size denote the time to download the entire file from single server, while the other bars show co-allocated downloads using all three servers. Our co-allocation strategy finished the jobs faster than the other strategies, and there was no combination time cost, and faster transmission than other co-allocation strategies.

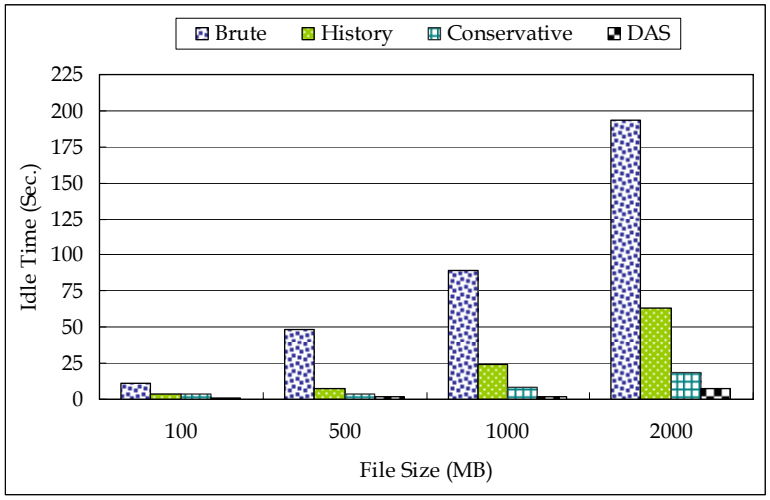

Fig. 3. Idle times for various methods; servers at PU, LZ, and HIT

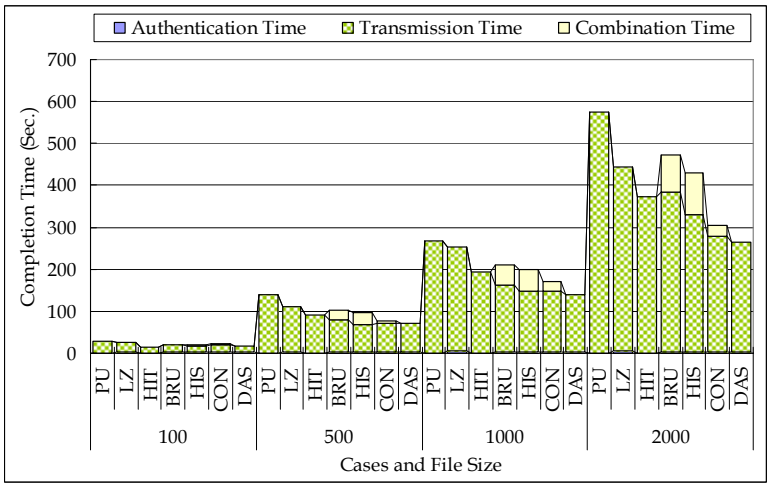

Fig. 4. Completion times for various methods; servers are at PU, LZ, and HIT.

5 Conclusions and Future Work

This paper proposes the Dynamic Adjustment Strategy to improve file transfer performances using the co-allocation architecture [17] in Data Grids. In our approach, the workloads on selected replica servers are continuously adjusted during data transfers, and our approach can also reduce the idle times spent waiting for the slowest servers, and thus decrease file transfer completion times.

We also developed a new toolkit, called Cyber-Transformer that enables even inexperienced users to easily monitor replica source site statuses, manage replicas, and download files from multiple servers in parallel. Experimental results show the effectiveness of our proposed technique in improving transfer times and reducing overall idle time spent waiting for the slowest servers.

In future work, we will investigate providing more functions for our user-friendly interface, for example, auto parameters input, and auto scan to find better replica servers for downloading. We also plan to improve replica management, especially on the problem of replica consistency.

References

1. B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data Management and Transfer in High-Performance Computational Grid Environments," Parallel Computing, 28(5):749-771, May 2002.
2. B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Secure, efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, pp. 13-28, 2001.
3. B. Allcock, S. Tuecke, I. Foster, A. Chervenak, and C. Kesselman. "Protocols and Services for Distributed Data-Intensive Science." ACAT2000 Proceedings, pp. 161-163, 2000.
4. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, and M. Ripeanu, "Giggle: A Framework for Constructing Scalable Replica Location Services," Proceedings of Supercomputing 2002, Baltimore, MD, 2002.
5. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Applications, 23:187-200, 2001.
6. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10'01), 181-194, August 2001.
7. K. Czajkowski, I. Foster, and C. Kesselman. "Resource Co-Allocation in Computational Grids," Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8'99), August 1999.
8. F. Donno, L. Gaido, A. Ghiselli, F. Prelz, and M. Sgaravatto, "DataGrid Prototype 1," Proceedings of the TERENA Networking Conference, http://www.terena.nl/conferences/tnc2002/Papers/p5a2-ghiselli.pdf, June 2002,
9. I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." International Journal of Supercomputer Applications and High Performance Computing, 15(3), pp. 200-222, 2001.
10. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," International Journal of Supercomputer Applications and High Performance Computing, 11(2), pp. 115-128, 1997.
11. Global Grid Forum, http://www.ggf.org/
12. W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data Management in an International Data Grid Project," First IEEE/ACM International Workshop on Grid Computing - Grid 2000, Bangalore, India, December 2000.
13. IBM Red Books, Introduction to Grid Computing with Globus, IBM Press, http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

14. H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney, "File and Object Replication in Data Grids," Journal of Cluster Computing, 5(3):305-314, 2002.
15. SYSSTAT utilities home page, http://perso.wanadoo.fr/sebastien.godard/
16. The Globus Alliance, http://www.globus.org/
17. S. Vazhkudai, "Enabling the Co-Allocation of Grid Data Transfers," Proceedings of Fourth International Workshop on Grid Computing, pp. 41-51, November 2003.
18. S. Vazhkudai and J. Schopf, "Using Regression Techniques to Predict Large Data Transfers," International Journal of High Performance Computing Applications (IJHPCA), 17:249-268, August 2003.
19. S. Vazhkudai and J. Schopf, "Predicting Sporadic Grid Data Transfers," Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 '02), pp. 188-196, July 2002.
20. S. Vazhkudai, J. Schopf, and I. Foster, "Predicting the Performance of Wide Area Data Transfers," Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), pp.34-43, April 2002.
21. S. Vazhkudai, S. Tuecke, and I. Foster, "Replica Selection in the Globus Data Grid," Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID 2001), pp. 106-113, May 2001.
22. R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," Future Generation Computer Systems, 15(5-6):757-768, 1999.
23. C.T. Yang, C.H. Chen, K.C. Li, and C.H. Hsu, "Performance Analysis of Applying Replica Selection Technology for Data Grid Environments," PaCT 2005, Lecture Notes in Computer Science, vol. 3603, pp. 278-287, Springer-Verlag, September 2005.
24. C.T. Yang, P.C. Shih, and S.Y. Chen, "A Domain-based Model for Efficient Network Information on Grid Computing Environments," accepted and to appear in IEICE Trans. Information and Systems, Special Issue on Parallel/Distributed Computing and Networking, vol. E89-D, no. 2, February, 2006, pp. 738-742.
25. C.T. Yang, S.Y. Wang, C.H. Lin, M.H. Lee, and T.Y. Wu, "Cyber-Transformer: A Toolkit for Files Transfer with Replica Management in Data Grid Environments," Proceedings of the Second Workshop on Grid Technologies and Applications (WoGTA'05), pp. 73-80, December 2005.
26. C.T. Yang, I.H. Yang, C.H. Chen, and S.Y. Wang, "Implementation of a Dynamic Adjustment Mechanism with Efficient Replica Selection in Co-Allocation Data Grid Environments," Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC 2006) - Distributed Systems and Grid Computing Track, pp. 797-804, April 23-27, 2006.
27. C.T. Yang, I.H. Yang, K.C. Li, and C.H. Hsu, "A Recursive-Adjustment Co-Allocation Scheme in Data Grid Environments," ICA3PP 2005 Algorithm and Architecture for Parallel Processing, Lecture Notes in Computer Science, vol. 3719, pp. 40-49, Springer-Verlag, October 2005.
28. C.T. Yang, I.H. Yang, K.C. Li, and S.Y. Wang, "Improvements on Dynamic Adjustment Mechanism in Co-Allocation Data Grid Environments," accepted and to appear in The Journal of Supercomputing, December 2006.
29. X. Zhang, J. Freschl, and J. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems", Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 '03), pp. 270-282, August 2003.
30. S. Venugopal, R.Buyya, and K. Ramamohanarao, "A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing", ACM Computing Surveys, 38(1):1-53, Mar-June 2006.