HIGH PERFORMANCE COMPUTING: PARADIGM AND INFRASTRUCTURE

HIGH PERFORMANCE COMPUTING: PARADIGM AND INFRASTRUCTURE

Edited by

L. Yang and M. Guo

A Wiley-Interscience Publication

JOHN WILEY & SONS

New York • Chichester • Weinheim • Brisbane • Singapore • Toronto

Contributors

XIAO QIN, Department of Computer Science, New Mexico Institute of Mining and Technology, 801 Leroy Place, Socorro, New Mexico 87801-4796

HONG JIANG, Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, Nebraska 68588-0115

Contents

Preface

Introduction

Part I

Cluster and Grid Computing

1 Data Grids: Supporting Data-Intensive Applications in Wide Area Networks

Xiao Qin* and Hong Jiang†

*Department of Computer Science
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro, New Mexico 87801-4796

†Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0115

1.1 INTRODUCTION

A grid is a collection of geographically dispersed computing resources, providing a large virtual computing system to users. The objective of a data grid system is two-fold. First, to integrate heterogeneous data archives stored in a large number of geographically distributed sites into a single virtual data management system. Second, to provide diverse services to fit the needs of high-performance distributed and data-intensive computing. There are four commonly used kinds of resources in grids: computation, storage, communications, and software. In what follows, we briefly introduced the storage resources.

Storage is viewed as the second most commonly used resource in a grid. The grid that presents an integrated view of storage is coined as "Data Grid". Memory, hard disk, and other permanent storage media are referred to as storage resources. In this study we are particularly interested in secondary storage systems in grids, since second storage is 1000 times slower than main memory attached to a processor. Throughout this chapter, we only address the issues of data grid with respect to secondary storage. Many Networked file systems, which exhibit security and reliability features, have been widely applied to data grid. These file systems include: Network File System

(NFS) [28] , Distributed File System (DFS), Andrew File System (AFS) [19], General Parallel File System (GPFS), and Parallel Virtual File System (PVFS) [16][44].

The amount of scientific data generated by simulations or collected from large-scale experiments is generally large, and such data tends to be geographically stored across wide-area networks for the sake of large-scale collaborations. The notion of computational grid has been proposed for several years, mainly focusing on effective usage of global computational and network resources in a wide area network/Internet environment. However, the performance of a computational grid, where the effective usage of storage resources is ignored, will be degraded substantially if vast majority of applications running in the grid are data-intensive. To overcome this problem, various techniques have been developed and incorporated into a so-called data grid infrastructure. In this chapter we will take a close look at these techniques presented in the literature, and point out some open issues in data grid research.

The rest of the chapter is organized as follows. Section 1.2 reviews three major data grid services. In Section 1.3, techniques for achieving high performance in data grids are described in detail. Security issue in data grid is briefly reviewed in Section 1.4. Section 1.5 identifies several open issues, and discusses the potential solutions to the open problems. Finally, Section 1.6 concludes the chapter by summarizing the main open problems and preliminary solutions.

1.2 DATA GRID SERVICES

In this section we focus on three essential servicesm: metadata service, data access service, and performance measurement.

1.2.1 Metadata Services for Data Grid Systems

The Metadata of a data grid system is the management information regarding to the data grid. The metadata includes file instances, the contents of file instances, and a variety of storage systems in the data grid. The goal of metadata service is to facilitate an efficient means of naming, creating and retrieving the metadata of data grid [13]. There are two types of metadata: application metadata and fabric metadata. While the former defines the logical structure of information represented by data files (e.g. XML [10]), the latter corresponds to information related to the characteristics of data grids.

Since a data grid consists of a large number of storage resources geographically dispersed over a large-scale distributed environment, an essential requirement of data grids is the capability of scaling up to accommodate a large number of users. Wahl et al. have proposed a so-called Lightweight Directory Access Protocol (LDAP) [41], which achieves a high scalability by applying a hierarchical naming structure along with rich data models. Fitzgerald et al. have developed a distributed directory service for general grid metadata, which is applied to represent the metadata of data grids [17]. Baru et al. proposed a metadata catalog (MCAT) for a storage resource

broker [4]. MCAT not only distinguishes logical name space from physical name space, but also provides the specification of a logical collection hierarchy, thereby achieving a high scalability. In addition, performance is improved in MCAT by introducing containers to aggregate small files.

Chervenak et al. recently developed a prototype Metadata Service (MCS) for data grids, where metadata is classified into two categories: logical and physical file metadata [14]. A logical file name in this prototype represents a unique identifier for a data content, whereas a physical file name denotes an identifier for a physical content on storage system. Unlike MCAT mentioned earlier, the services in MCS are dedicated to describing files, and the physical file properties are not covered by MCS. While MCAT utilizes a proprietary data exchange format to communicate with storage resource broker servers, MCS employs XML interface [10] to communicate with clients.

1.2.2 Data Access Services for Data grid Systems

Data stored and retrieved by a data grid may reside, by design, in different sites on different storage devices. For this reason, data grids have to provide a global view of data for applications running on the grids [13]. Data access services are required to be compatible with traditional file systems and, as a consequence, applications that are not originally designed for grid environments can run on a grid without complicated modifications.

Globus, serving as an infrastructure for grid computing, provides access to remote files through x-gass, ftp or HTTP protocols [18]. White et al. have proposed the Legion I/O model that provides a remote access capability [42]. To allow end users to exploit context space and manipulate context structures, Legion also offers some command line utilities, which are similar to those in a Unix File System. Patten and Hawick have proposed a Distributed Active Resource Architecture (DARC) that enables the development of portable, extensible, and adaptive storage services optimized for end users' requirements [24].

1.2.3 Performance Measurement in Data Grid

Since multiple resources may affect one another across time and space, performance problems (e.g. low throughput and high latency) might be imposed by a combination of resources. Thus, it is important and non-trival to diagnose performance problems in accordance with performance measurements provided by monitors. With a monitoring service in place, one can analyze the monitoring data from different point of views. The objective of the monitoring service is to observe performance bottlenecks, which are likely to occur in any components of the data grid.

Previous studies have demonstratively shown that monitoring at application level is an effective approach to both performance analysis and application debugging [9][39]. For example, Snodgrass has incorporated relational databases into a monitoring system to monitor complex systems [29].

GMA (Grid Monitoring Architecture) embraces the key components of a grid monitoring system along with some essential interactions [38]. The data model used in the Network Weather Service [34] has been extended by Lee et al. to fulfil the special needs of monitoring data archives in grid environments [21]. More specifically, Lee et al. has proposed a relational monitoring data archive that is designed to efficiently handle high volume streams of monitoring data [21].

1.3 HIGH PERFORMANCE DATA GRID

We now turn our attention to the issues related to data replication, scheduling, and data movement, which must be addressed in order to achieve high performance and scalability in data grids.

1.3.1 Data Replication

A large body of work has been found on data replication for distributed applications [11][32][35]. The existing techniques supporting data replication in data grids can be divided into four camps, namely, (1) Architecture and management for data replication, (2) Data replication placement, (3) Data replication selection, and (4) Data consistency.

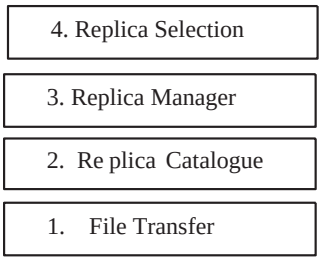

Fig. 1.1 Architecture of a data Replication Service in a data grid system.

Architecture of a data replication service
A general high-level architecture with four layers for data replication services of a data grid system is depicted in Figure 1.1 [13]. We describe each layer as follows:

1. File transfer layer consists of file transfer protocols such as: FTP, HTTP, and GridFTP [3]. Note that GridFTP is an extended version of FTP.

2. The replica catalogues [3] layer is leveraged to map logical file names to physical files.

3. The replica manager layer's functionalities include create, move, modify and delete replicas in a data grid. Once a file is copied from one site to another by

the replica manger layer and registered in the replica catalogue, the file can be retrieved by any application running on arbitrary site in the grid.

4. The replica selection layer is responsible for choosing the most appropriate replica from those copies geographically dispersed across grids. The selections of replication policies largely depend on read and write accesses to data that can be classified into the three categories: Read-only data, writable data with well-defined file ownership, and writable data with varying writers.

Data Replication Management

We first take a look at replication policies designed for read-only data. A pilot project called the Grid Data Management Pilot (GDMP) has been launched to develop a prototype for replication management within a data grid environment [27]. In the GDMP software, a secure replication of database files has been used, and a high-level replica catalog interface is implemented by using the native objectivity federation catalogue.

More broadly, replications can be optionally implemented at different levels of granularity, such as file level and object level [31]. Realizing that file replications in a data analysis application are potentially inefficient, Stockinger et al. have fully implemented object replication in a data grid prototype using Globus Data Grid tools [31].

Foster et al. recently developed a general framework, which defines a parameterized set of basic mechanism to create a variety of replica location services [15]. By configuring system parameters dynamically, one is able to tune system performance by means of considering the tradeoff among reliability and communication overhead, and storage update/access costs.

Data Replication Placement

In a recent study [25], several replication selection strategies for managing large data set have been proposed and evaluated. Unlike static replications, dynamic replication approaches automatically add and delete replicas in accordance with changing workload, thereby sustaining high performance even in the face of diverse data access patterns [25].

Data Replication Selection

Once data replicas are generated and stored at multiple sites according to a variety of replication placement strategies discussed above, the performance of data grid can be noticeably improved by choosing optimized replica location for each data access.

Replica selection is a high level service that helps grid applications to choose a replica based on system performance and data access features. Vazhkudai et al. [40] have designed and implemented a replica selection service that uses information with respect to replica location as well as application's specific requirements to determine a replica among all replica alternatives.

Data Consistency in Data Grid Systems

The data replication techniques reviewed above mainly aim at maintaining replication for read-only files, thereby avoiding data consistency problems. In general, local consistency at each site is completely guaranteed by database management systems (DBMS), whereas global consistency in data grid systems needs to be maintained by a so-called grid consistency service.

There have been extensive studies of data consistency in distributed environments [33][43]. The main target of the consistency model proposed by Sun et al. [33] is real-time cooperative editing systems that allow multi-user, physically dispersed people, to view and edit a shared document at the same time over networks. To address the issue of grid data consistency maintenance, Düllmann et al. proposed a consistency service [23], which is supported by existing data grid services.

To maintain the basic consistency level, a replication operation must be manipulated as a database read transaction [23]. Thus, a read lock on a file has to be obtained before a site is ready to produce a new replica of the file. The site releases the read lock when the replica has been successfully duplicated. Likewise, the consistency service rests on write locks when update operations are issued on a set of replicas. All data replication techniques discussed above are summarized in Table 1.1.

Table 1.1 Summary of Data Replication Techniques

Researchers	Granularity	Data Type	Consistency	Technique
Samar et al.	File	Read Only	No[a]	Subscription and partial replication model
Vazhkudai et al.	File	Read Only	No	Replication selection
Stockinger et al.	File/Object	Read Only	No[b]	Object replication
Chervenak et al.	File	Read Only	Yes[c]	Replication location service
Ranganathan et al.	File	Read Only	No	Replication placement
Lamehamedi et al.	File	Read Only	No	Replication placement
Mullmann et al.	File	Read/Write	Yes	Consistency service
Sun et al.	String	Read/Write	Yes	Three novel properties

[a] Producer decides when to publish new files.
[b] Consumer decides when to replicate objects.
[c] Relaxed consistency for metadata.

1.3.2 Scheduling in Data grid Systems

Previous research has shown that scheduling is a fundamental approach to achieving good performance for grid applications [1][7] [30][35]. However, these approaches do not take into account distributed data constraints.

Scheduling schemes designed for data-intensive jobs have generally been classified into two major categories: moving data to job and moving job to data. To reduce the frequency of remote data access and the amount of transferred data, data replication can be an optimization technique as we discussed in Section 1.3.1.

Thain et al. developed a system where execution sites band together into I/O communities [36]. Each I/O community comprises several CPUs and storage devices that provide storing and retrieving data services both for local and remote jobs. To enable jobs to declare constraints on storage devices within I/O communities, Thain et al. extended ClassAds [36] that are currently used in Condor system [22] to describe jobs' requirements in a distributed system. A single ClassAd is a list of (attribute, value) pairs, where value can be either atomic or complex expressions. Figure 1.2 illustrates three different kinds of ClassAd [36].

```
Type = "job"                Type = "machine"    Type = "storage"
TargetType = "machine"      TargetType = "job"  Name = "machine"
Cmd = "sim.exe"             Name = "raven"      HasCMSData = true
Owner = "thain"             Opsys = "linux"     CMSDataPath =
Requirements =              Requirements =              "/cmsdata"
 (Owner = "thain") &&         (Onwer = "thain")
NearestStorage.HasCMSData   NearestStorage =
                              (Name="turkey")&&
                              (Type = "storage")
```

Fig. 1.2 Examples of job ClassAd, Machine ClassAd, and Storage ClassAd.

To minimize remote data access overheads, Basney et al. have invented a so-called execution domain framework to define an affinity between CPU and data resources in the grid [5]. By applying the framework to the Condor system, data-intensive jobs can be scheduled to run on CPUs that have access to required data.

To gain an efficient execution of parameter sweep applications on the grid, Casanova et al. have studied four adaptive scheduling algorithms that consider distributed data storage [12]. The proposed algorithms are heuristic yet effective in environments where some computing nodes are best for some tasks but not for others. The key idea behind these scheduling algorithms is to judiciously place files for maximum reuse. These scheduling algorithms have been implemented in a user-level grid middleware, which centrally handles task scheduling.

Ranganathan and Foster have proposed a distributed scheduling framework where each site consists of an external scheduler, a local scheduler, and a dataset scheduler [26]. The external scheduler of a site is responsible for dispatching the submitted jobs to an appropriate site, which can either be one with the least load or the one where required data have been staged. The functionality of the local scheduler is to decide the order in which jobs are executed at the site. The dataset scheduler makes decisions on when and where to replicate data.

1.3.3 Data Movement

In a computational grid, data movement mechanisms play an important role in achieving high system performance. This is driven by two requirements imposed by both grid applications and data grid itself. First, applications need to access data that is not located at the site where the computation is performed. Second, replication services are responsible for managing replicas by means of copying and moving data through networks.

To fulfill the above two requirements, Bester et al. have proposed a data movement and access service called Global Access to Secondary Storage (GASS) [8]. The service not only incorporates data movement strategies that are geared for the common I/O patterns of grid applications, but also supports programmer management of data movement. The performance evaluation shows that GASS is an efficient approach to data movement in grid environments.

Thain et al. have developed a data movement system called Kangaroo, which improves reliability as well as throughput of grid applications by hiding network storage devices behind memory and disk buffer [37]. More specifically, the approach relies on Kangaroo's ability to overlap CPU and I/O processing intervals by using background processes to move data and handle errors.

1.4 SECURITY ISSUE

With the usage of different machines in a Grid, the issue of security, which has been provided by most existing data grids [13][27][42], becomes one of the prime concerns. In principle, traditional security techniques, such as encryption and access control, can be conservatively applied to a grid. For example, a user has to be authenticated and authorized before contacting any remote site.

Due to the space limits, in what follows, we only review two intriguing issues in security addressed in GDMP [27], namely, the main security issues of sensitivity of data and unauthorized use of network bandwidth to transfer huge files. Specifically, the security module of GDMP server, based on the Globus Security Service (GSS) API, offers functions to acquire credentials, initiate context establishment on client side and accept context requests on server side, encrypting and decrypting messages and client authorization. Consequently, the security module protects a site from any undesired file transfers and rejects any unauthorized requests.

1.5 OPEN ISSUES

Although a variety of techniques have been proposed to achieve high performance in data grids, there are still some open issues to be addressed. This section lists the open issues related to high performance data grid.

1.5.1 Application Replication

Although some efforts have been made to replicate actual data accessed by grid applications, little attention has been paid to replicating frequently used grid applications across the grid. Once a scheduling scheme decides to move an application towards its data, a replication selection policy for applications can be developed to choose the most appropriate application copy in the grid. There are three reasons that motivate us to distinguish application code from actual data:

- Foremost, an executable code of an application at one site might not be able to run on another site. One can straightforwardly tackle this problem by using a java-programming environment, which becomes increasingly popular [2]. However, a large number of existing scientific applications have been developed in FORTRAN, C or other programming languages. To solve this problem, one can have the source code moved to destination sites and obtain the executable code by compiling the source code at the destination sites.

- The application replication techniques provide efficient support to a parallel-shared-nothing architecture. For example, a well-designed partition algorithm can be used to dynamically and effectively divide data among the processors where the processing application has been staged to achieve maximum parallelism.

- It is reasonably assumed that data can be stored at any site, provided that it has registered as a member of a data grid. Unfortunately, this assumption is not practical for applications since some sites might not be able to fulfill applications' hardware or/and software requirements.

To facilitate a parallel computing environment in computational grids, one of the future directions can be studying an application replica service, which includes application replica management and application selection. The application replica management is designed to duplicate, remove, and update copies of application instances, as well as to maintain the consistency among the replicas. The application selection service is devised to choose an optimized location where an application replica has been stored to run the job.

1.5.2 Consistency Maintenance

In Section 1.3.1, we have discussed the requirements of consistency maintenance imposed by scientific data and metadata. We believe that the proposed application replication services make the consistency maintenance even more complex. This is because in order to obtain high scalability, it is appealing to offer the services in a distributed fashion. In addition, a group of programmers are allowed to collaborate with one another to develop some large-scale applications in a grid environment, and grid applications are likely to be periodically upgraded.

We realize that from the summary of data replication techniques given in Table 1.1, most existing consistency models in data grid work well for read only data, but much less attention has been devoted to the consistency issues for data that needs to be written multiple times. In a data grid environment, many datasets are multiple-written in nature. When a group of programmers collaboratively develop a large-scale scientific applications on the grid, some programmers might be working on the same source code, implying that the code not only needs to be written multiple times but also likely to be written concurrently by several programmers. Therefore, it will be interesting to develop a relaxed consistency model to effectively handle the consistency problems in multiple-written data discussed above.

1.5.3 Asynchronized Data Movement

The impact of data movement on the performance of local site is enormous, and this is especially true if the volume of transferred data is huge or data movements occur frequently. To alleviate such a burden resulting from data movements, a future direction is to study a new way of moving data without sacrificing the performance of applications running on local sites. In particular, a protocol can be designed to move asynchronized data if the load of source site and destination site is below a certain threshold. Compared with existing data movement approaches that treat synchronized and asynchronized equally, our potential solution is expected to improve the performance of data grids when I/O and network load at each site is bursting in nature.

1.5.4 Prefecthing Synchronized Data

We notice that data prefetching schemes for synchronized data have caught little attention. We believe that data prefetching technique, which complements the new data movement approaches presented in Section 1.3.3, can optimize the performance of data grids in case where required data is moved towards to a remote site where its job resides.

By using ClassAd language to explicitly define applications' required files, a static prefectching scheme can make these files available before applications need to access them. As one of the possible future directions, a new prefetching algorithm, which is able to dynamically predict files, will be developed to establish correlations among files based on a statistic model. This new dynamic prefetching algorithm actively provides grid applications with files by shipping required files in advance before applications are locally loading the files.

1.5.5 Data Replication

As can be seen from Table 1.1, the idea of replicating frequently accessed data on multiple sites has been eagerly pursued. The drawback of most existing replication techniques is that as replication granularity is file, a large file might be replicated even

if only a small portion of the file is frequently accessed. This problem is analogous to false sharing in distributed shared memory system. To provide a potential solution to tackle this problem, one can investigates a new data model, where the replication granularity can be dynamically adjusted according to benefits gained from replicas and overheads of generating replications.

Interestingly, there is no clear consensus on developing caching algorithms with efficient disk cache replacement policies in the context of data grids. As a result, there is a distinct need to study a caching algorithm, in which files cached on local disks are viewed as temporary replicas. The main difference between cached files and regular replicas is that files in disk cache might be removed when disk cache is unable to accommodate newly arrived files.

1.6 CONCLUSIONS

We have reviewed a number of data grid services and various techniques for improving the performance of data grid. Additionally, the issue of security in data grid has been briefly discussed. Five open issues and the possible future directions are summarized as below.

1. While a significant amount of data grid research has been done in data replication, the idea of replicating frequently used grid applications across the grid has received little attention. One possible future work is to develop a new application replica service, which can be built on top of data replication service.

2. It is noticed that most consistency models in existing data grids have not addressed consistency issues for multiple-written data. As one of the future directions, one can develop a relaxed consistency model to effectively handle the consistency problems for a number of common file access patterns including multiple-written pattern.

3. There is a distinct need to study a new way of moving data at a time when both source and destination sites are lightly loaded, thereby achieving better performance in data grid without sacrificing the performance of applications running on local sites.

4. A fourth interesting future direction is to design a new prefetching algorithm that can dynamically predicate files that are likely to be retrieved.

5. Last but not least, one of the possible future directions is to study a file-caching algorithm, where the replication granularity of a new data model can be dynamically tuned in accordance with the benefits and overheads of generating replicas.

Acknowledgments

This work was partially supported by an NSF grant (EPS-0091900), a Nebraska University Foundation grant (26-0511-0019), and a UNL Academic Program Priorities Grant.

REFERENCES

1. D. Abramson, J. Giddy, I. Foster, and L. Kotler, "High Performance Parametric Modeling with Nimrod /G: Killer Application for the Global Grid," *Proc. of the International Parallel and Distributed Processing Symposium,* May, (2000).

2. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "An Agent-Based Infrastructure for Parallel Java on Heterogeneous Clusters", *Proc. of 4th IEEE International Conference on Cluster Computing,*Chicago, Illinois, September (2002).

3. B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, et. al., "Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," *Proc. 18th IEEE Symposium on Mass Storage Systems,* (2001).

4. C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker," *Proc of CASCON Conference,* (1998).

5. J. Basney, M. Livny, and P. Mazzanti, "Utilizing Widely Distributed Computational Resources Efficiently with Execution Domains", *Computer Physics Communications*, Volume 140, (2001).

6. J.Basney, R.Raman, and M.Livny, "High-throughput Monte Carlo," *Proc. ot the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March (1999).

7. F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks," *Proc. of Supercomputing,* (1996).

8. J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," *Proc. of Sixth Workshop on I/O in Parallel and Distributed Systems*, May 5, (1999).

9. W. Bethel, B. Tierney, J. Lee, D. Gunter, and S. Lau, "Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization," *Proc. of the IEEE Supercomputing Conference*, (2002).

10. T. Bray, J. Paoli, and C. Sperberg-McQueen, The Extensible Markup Language (XML)1.0 W3C Recommendation, *World Wide Web Consortium*, Feb. (1998).

11. L. Breslau, P. Cao, L, Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *Proceedings of IEEE Infocom*, (1999).

12. H. Casanova, G. Obertelli, F. Berman, and R. Wolski, "The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid," *Proc. Supercomputing*, (2000).

13. 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," *Proc. of Network Storage Symposium*, (2000).

14. A. Chervenak, E. Deelman, C. Kesselman, L. Pearlman, and G. Singh, "A Metadata Catalog Service for Data Intensive Applications", *GriPhyN Technical Report*, Information Science Institute, (2002).

15. A. Chervenak, E. Deelman, I. Foster, et al., "Giggle: A Framework for Constructing Scalable Replica Location Services," *Proc. IEEE Supercompuing Conference*, (2002).

16. A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp, "Noncontiguous I/O through PVFS," *Proceedings of IEEE International Conference on Cluster Computing*, September, (2002).

17. S. Fitzgerald, I. Foster, C. Kessenlman, G. Laszewski, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-performance Distributed Computations," *Proc. 6th IEEE Symposium on High Performance Distributed Computing*, pp.365-375, (1997).

18. I. Foster and C. Kesselman, "Globus: A Metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, Vol.11, No.2, pp.115-128, (1997).

19. J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, "Scale and Performance in a Distributed File System," *ACM Transactions of Computer Systems*, Vol6, No.1, pp.51-81, (1988).

20. H. Lamehamedi, B. Szymanski, Z. shentu, and E. Deelman, "Data Replication Strategies in Grid Environments," *Proc. Of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*, (2002).

21. J. Lee, D. Gunter, M. Stoufer, and B. Tiemey, "Monitoring Data Archives for Grid Environments," *Proc. Supercomputing*, (2002).

22. M. Litzkow, M. Livny, and M. Mutka, "Condor-A Hunter of Idle Workstations," *Proc. of the 8th International Conference on Distributed Computing Systems*, pp.104-111, (1988).

23. D. Mullmann, W. Hosckek, J. Jaen-Martinez, and B. Segal, "Models for Replica Synchronization and Consistency in Data Grid," *Proc. IEEE Symp. on High Performance on Distr. Computing*, (2001).

24. C.J. Patten and K.A. Hawick, "Flexible High-performance Access to Distributed Storage Resources," *Proc of High Performance Distributed Computing*, (2000).

25. K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," it Proc. International Grid Computing Workshop, Denver, November (2001).

26. K. Ranganathan and I Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications," *Proc. of the 11th IEEE Symposium on High Performance Distributed Computing*, (2002).

27. A. Samar, H. Stockinger, "Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication," *Proc. of International Conference on Applied Informatics*, Feb. (2001).

28. R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem," *Proc. of USENIX Conference*, Berkeley, (1985).

29. R. Snodgrass, "A Relational Approach to Monitoring Complex System," *ACM Transactions on Computer Systems*, Vol.6, No.2, pp.157-196, (1988).

30. N. Spring and R. Wolski, "Application level scheduling: Gene sequence library comparison," *Procc of ACM International Conference on Supercomputing*, (1998).

31. H. Stockinger, A. Samar, B. Allock, I. Foster, K. Holtman, and B. Tierney, "File and Ojbect Replication in Data Grids," *Proc. of IEEE Symp. on High Performance Distributed Computing*, (2001).

32. H. Stockinger, "Distributed Database Management Systems and the Data Grid," *18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, April 17-20, (2001).

33. C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, "Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems," *ACM Transactions on Computer-Human Interaction*, Vol.5, No.1, pp.63-108, March, (1998).

34. M. Swany and R. Wolski, "Representing Dynamic Performance Information in Grid Environments with Network Weather Service," *Proc. of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, May (2002).

35. R. Tewari, M. Dahlin, H. Vin, and J. Kay, "Design Considerations for Distributed Caching on the Internet," *Proc. of IEEE International Conference on Distributed Computing Systems,* (1999).

36. D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny, "Gathering at the Well: Creating Communities for Grid I/O," *Proc. Supercomputing,* (2001).

37. D. Thain, J. Basney, S. Son, and M. Livny, "The Kangaroo Approach to Data Movement on the Grid," *Proc. of the 10th IEEE Symposium on High Performance Distributed Computing,* (2001).

38. B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany, "A Grid Monitoring Service Architecture," *Global Grid Forum While Paper,* (2001).

39. B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, and D. Gunter, "The NetLogger Methodology for HIgh Performance Distributed Systems Performance Analysis," *Proc. of IEEE High Performance Distributed Computing,* July, (1998).

40. S. Vazhkudai, S. Tuecke, and I. Foster, "Replica Selection in the Global Data Grid," *Proc. of the International Workshop on Data Models and Databases on Clusters and the Grid,* (2001).

41. M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol," *RFC 2251, Internet Engineering Task Force,* (1997).

42. B. White, A.S.Grimshaw, and A. Nguyen-Tuong, "Grid-Based File Access: The Legion I/O Model," *Proc. of High Performance Distributed Computing,* (2000).

43. H. Yu and A. Vahdat, "Design and Evaluation of a Conit-based Continuous Consistency Model for Replicated Services," *ACM Transactions on Computer Systems,* August (2002).

44. Y. Zhu, H. Jiang, X. Qin, and D. Swanson, "A Case Study of Parallel I/O for Biological Sequence Analysis on Linux Clusters", *Proceedings of the 5th IEEE International Conference on Cluster Computing,* pp.308-315, Hong Kong, Dec. 1-4, (2003).