AN IDA-BASED PARALLEL STORAGE SCHEME IN THE SCIENTIFIC DATA GRID

Weizhong Lu[1,2*], Yuanchun Zhou[1], Lei Liu[1,3], and Baoping Yan[1]

[1]*Computer Network Information Center, Chinese Academy of Sciences, Beijing, China*
**Email:* lvweizhong@cnic.cn
[2]*Graduate University of Chinese Academy of Sciences, Beijing, China*
[3]*CODATA-China Secretariat, Beijing, China*

ABSTRACT

It is important to improve data reliability and data access efficiency for data-intensive applications in a data grid environment. In this paper, we propose an Information Dispersal Algorithm (IDA)-based parallel storage scheme for massive data distribution and parallel access in the Scientific Data Grid. The scheme partitions a data file into unrecognizable blocks and distributes them across many target storage nodes according to user profile and system conditions. A subset of blocks, which can be downloaded in parallel to remote clients, is required to reconstruct the data file. This scheme can be deployed on the top of current grid middleware. A demonstration and experimental analysis show that the IDA-based parallel storage scheme has better data reliability and data access performance than the existing data replication methods. Furthermore, this scheme has the potential to reduce considerably storage requirements for large-scale databases on a data grid.

Keywords: Parallel, Storage, Data Grid

1 INTRODUCTION

There are increasing requirements to process very large datasets in many high performance and data intensive application domains. Many scientific research projects are generating petabytes of experimental data every year. Processing and sharing such huge amounts of data among collaborative institutes poses new challenges to the parallel computing community and data storage. The two important challenges that data-intensive applications face are virtual massive data storage and high data access efficiency, which can be solved by introducing grid technology and data parallel storage schemes, respectively.

Grid computing is an emerging technology to connect geographically distributed computing resources with storage resources to provide high performance systems for users from different areas. The Scientific Data Grid of the Chinese Academy of Sciences is a fundamental infrastructure for many data-intensive natural scientific research projects. It provides a large shared storage environment for the geographically distributed and heterogeneous multidisciplinary data resources from different institutes. Data replication is an effective way to improve data availability and data access efficiency. However, recent research work shows that the most effective method adopts distributed storage and a parallel access mechanism to get high data availability and high data transport throughput (Allcock, Bresnahan, Kettimuthu, Link, Dumitrescu, & Raicu, et al., 2005; Ghemawat, Gobioff, & Leung, 2003; Raicu, Zhao, Foster, & Szalay, 2008).

There are several advantages to using distributed storage and parallel access technology for data-intensive applications in a data grid environment. First of all, because a data file is cut into blocks that are distributed among grid nodes, parallel access can easily adapt to the changing network conditions. Secondly, when integrating some fault tolerance algorithms to implement a reliable system, parallel access, by performing automatic load balancing, is more resistant to congestion and failure in the network and servers than is connecting to a single server. Thirdly, high data access performance can be achieved. Ideally, the total throughput seen by one client is equal to the sum of the bandwidth from each individual storage server to the client.

In this paper, we propose an IDA-based parallel storage scheme, which takes high data availability and data access efficiency into consideration, in the scientific data grid. In this scheme, every data file is encoded into a number of blocks that can be spread across the storage nodes in the grid. The data file can be reconstructed from any of the blocks with simultaneous parallel access to the selected blocks. Thus we can achieve better data availability and data access efficiency than the usual data replication technologies.

The paper is organized in the following way. In section 2, some related work is introduced, and the motivation for our method is given. In section 3, our system model and algorithm are described in detail. Section 4 contains the experiment and analysis. Section 5 concludes this paper.

2 RELATED WORK

Data grids are playing a more and more important role in sharing large data collections throughout the scientific community. In recent years, many data storage and access technologies have been developed in the data grid research area to improve data access reliability and performance. In this section, we investigate several data storage and access schemes, such as Globus replication, GridFTP, SRB, iRODS, Google File System, MapReduce, and Grid File System.

The open source Globus Toolkit (http://www.globus.org/) is a fundamental enabling technology for the grid that allows people to share computing power, databases, and other tools securely across corporate, institutional, and geographic boundaries, without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management. To get high data availability, the Globus Toolkit introduced and implemented a Data Replication Service (DRS) that provides a pull-based replication capability for grid files. Therefore, data replication is a very important method to improve data availability in the Globus Toolkit grid environment. Another important component of the Globus Toolkit for high data access performance and reliable massive data file transport is GridFTP (Allcock, Bresnahan, Kettimuthu, Link, Dumitrescu, Raicu, et al., 2005). GridFTP, based on FTP, is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. GridFTP achieves a much greater use of bandwidth by allowing multiple simultaneous TCP streams. Files can be downloaded in pieces simultaneously from multiple sources or even in separate parallel streams from the same source.

The Storage Resource Broker (SRB) (Baru, Moore, Rajasekar, & Wan, 1998) and the Integrated Rule-Oriented Data System (iRODS) (http://www.irods.org/) are two important, widely adopted data grid software developed by the San Diego Supercomputing Center (SDSC). SRB, which is considered to be the first generation data grid system, is a storage management system designed for data grid environments. SRB is client-server middleware that provides a uniform interface and mechanism to access heterogeneous data resources distributed in multiple hosts and multiple platforms by using logical to physical name mapping. iRODS, which is extended from SRB, is a second generation data grid system providing a unified view and seamless access to distributed digital objects across a wide area network. While SRB focuses mainly on providing a unified view over distributed storages based on logical naming concepts using the client server architecture, iRODS takes things one level higher by abstracting the data management process itself through what is called policy abstraction. In iRODS, rules are explicitly declared to control the operations performed when a rule is invoked by a particular task. iRODS implements data file replication and parallel transport by executing the rules. Ingestion of data files into the iRODS digital system is accomplished by using the iPUT command. Just like GridFTP, the iPUT command is a multithreaded application and adapts the number of threads it uses to the size of the file it transfers.

Another large data management and processing system is Google's MapReduce system (Dean, & Ghemawat, 2008), which runs on the top of the Google File System (GFS) (Ghemawat, Gobioff, & Leung, 2003). In the Google File System, a data file is loaded, partitioned into chunks, and each chunk is replicated. Thus data processing is collocated with data storage. When a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk and then schedules a "map" process on that node, so that data locality is exploited efficiently. MapReduce, along with Google File System and BigTable, couples data resources and computing resources to accelerate data-intensive applications. In short, these systems achieved distributed storage and parallel transport of data chunks based on GFS and replication technology.

The last grid data management system is the Grid File System (García-Carballeira, Carretero, Calderón, García, & Sanchez, 2007). The Global Grid Forum defines a Grid File System as a human-readable resource namespace for management of heterogeneous and distributed data resources that can span across multiple autonomous administrative domains. García-Carballeira, Carretero et al. (2007) described a new Grid File System according to the Global Grid Forum recommendations that integrates heterogeneous data storage resources in grids using standard grid technologies: the GridFTP and the Resource Namespace Service, both defined by the Global Grid Forum. To obtain high performance, the parallel transmission techniques used in traditional parallel file systems are adopted. Although the Grid File System implements data file parallel storage and access, it does not actually improve data file reliability for any failure of any file chunks would destroy the whole data file.

From all of the above, data replication and parallel transport are practical and effective methods of achieving efficient data access in a grid environment. Globus and iRODS implement data file replication and achieve high data file transport performance by GridFTP and iCommands, which establish multi-connections between each source and destination. The Google MapReduce framework and the Grid File System both introduce distributed storage technology to achieve high data file access performance by transferring data files in a parallel way. In contrast, this paper proposes a new parallel storage scheme in which a data file is encoded into blocks and distributed on different storage servers. It achieves high data availability and high parallel data access efficiency physically at the same time.

3 THE IDA-BASED PARALLEL STORAGE SCHEME

3.1 System Overview

In this section, we introduce the IDA-based data parallel storage scheme and its components in detail. The scheme basically focuses on data availability and parallel access. Currently, the most common method of achieving high data availability is data replication technology, which uses several times as much as storage space as does our scheme. In contrast, the IDA-based parallel storage scheme distributes data file blocks across different storage servers achieving better data availability and can support the parallel access mechanism for high performance at the same time.

One of the key technologies of the IDA-based parallel storage scheme is the information dispersal algorithm (Rabin, 1989). The information dispersal algorithm was proposed as a fault-tolerance technique to address the security and reliability problems in massive data storage systems. In the IDA algorithm, a data file F is striped into n blocks of size L/m, where L is the length of the data file and m is the number of blocks required to reconstruct the data file F. The diagram of the IDA-based parallel storage scheme is shown in Figure 1.

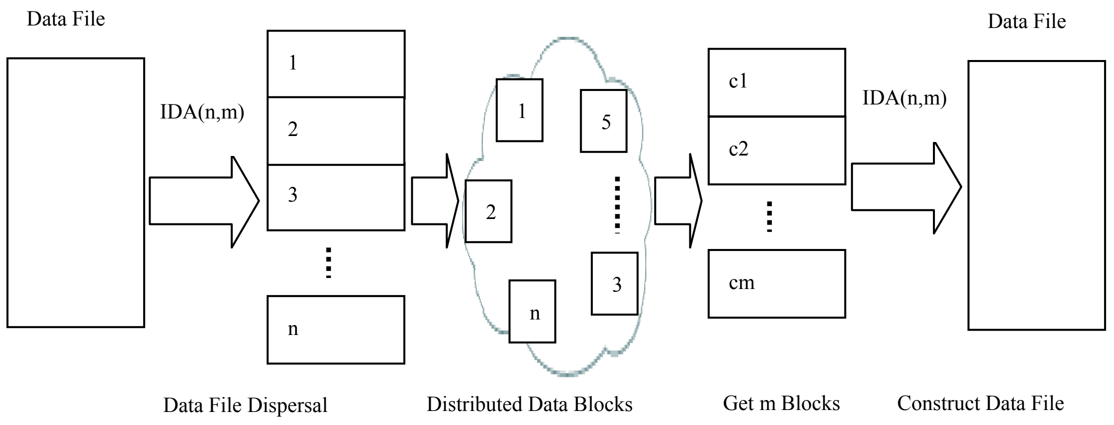


Figure 1. The diagram of information dispersal algorithms

A set of secret keys are used to disperse the file, providing confidentiality to the information at the same time. Since $m \leq n$, the redundancy level given by (n/m-1) % can be selected to be smaller than the replication technique. The storage requirement is $L*(n/m)$. Additionally, the information dispersal algorithm tolerates up to f failures, where $f = n - m$, and guarantees a higher availability.

3.2 The Information Dispersal Algorithm

The processing of data within the information dispersal algorithm includes two phases: data file dispersal and data file reconstruction. In the data file dispersal phase, a data file is divided into blocks and encoded into encrypted files respectively, which will be dispersed into the distributed systems according to applications. In the data file reconstruction phase, a subset of the encrypted files, needed to reconstruct the original data file, is retrieved.

In the first phase, to disperse a data file F, a matrix $A_{n \times m}$ must be chosen. The matrix, as shown in Equation 1, is composed of a set of vectors ($V_1, V_2, V_3, \ldots, V_n$), each of which has length m. These vectors are the keys that will be used to encode the blocks of the original data file and recover the original data file from the blocks. One important requirement of the vectors is that any subset of m different vectors is linearly independent. We define the length of the data file to be L; then r equals L/m denotes the number of file blocks (Equation 2). The data file is divided into sequences of blocks ($F_1, F_2, F_3, \ldots, F_r$) with length m. Accordingly, the data file F can be described by a matrix $F_{m \times r}$, as in Equation 3. Then the dispersal operation is achieved by multiplying matrices

$A_{n\times m}$ and $F_{m\times r}$, and the product, matrix $C_{n\times r}$, is the ciphered data file. The detailed processing is shown in Equations 4 and 5. Finally, the output of the dispersal phase, the matrix $C_{n\times r}$ will be stored in *n* separate block of files with each row of the matrix being stored as one file ($C_1, C_2, C_3, ..., C_n$).

$$A_{n\times m} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \tag{1}$$

$$r = \frac{L}{m} \tag{2}$$

$$F_{m\times r} = \begin{bmatrix} F_1 & F_2 & \ldots & F_r \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1r} \\ f_{21} & f_{22} & \cdots & f_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mr} \end{bmatrix} \tag{3}$$

$$A_{n\times m} \cdot F_{m\times r} = C_{n\times r} \tag{4}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \cdot \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1r} \\ f_{21} & f_{22} & \cdots & f_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mr} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1r} \\ c_{21} & c_{22} & \cdots & c_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nr} \end{bmatrix} \tag{5}$$

In the second phase, in order to reconstruct the original data file *F*, *m* block files and the matrix $A_{n\times m}$ are required. Given *m* scattered blocks ($C_{s1}, C_{s2}, C_{s3}, ..., C_{sm}$) that can be expressed by a matrix $C_{m\times r}$, we know that for each block C_{si} ($1 \le i \le m$) is the product of V_{si} ($1 \le i \le m$) and $F_{m\times r}$ according to Equations 4 and 5. We let $B_{m\times m}$ be a matrix whose rows are vectors ($V_{s1}, V_{s2}, V_{s3}, ..., V_{sm}$) from matrix $A_{n\times m}$, as shown in Equation 6. The matrix $C_{m\times r}$ can be expressed by the result of multiplication of matrix $B_{m\times m}$ and matrix $F_{m\times r}$, as shown in Equation 7. Consequently, the original data file $F_{m\times r}$ is recovered by left multiplication of the inverse of the matrix $B_{m\times m}$ (B^{-1}) and $C_{m\times r}$, as shown in Equation 8.

$$B_{m\times m} = \begin{bmatrix} V_{s1} \\ V_{s2} \\ \vdots \\ V_{sm} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mm} \end{bmatrix} \tag{6}$$

$$B_{m\times m} \cdot F_{m\times r} = C_{m\times r} \tag{7}$$

$$B_{m\times m}^{-1} \cdot B_{m\times m} \cdot F_{m\times r} = F_{m\times r} = B_{m\times m}^{-1} \cdot C_{m\times r} \tag{8}$$

It should be noted that obtaining the inverse of matrix $B_{m\times m}$ is always guaranteed because the rows of the matrix $A_{n\times m}$ are mutually independent, which implies that any submatrix of $A_{n\times m}$ is nonsingular and thus invertible.

3.3 The IDA-based Data Parallel Storage System

The IDA-based data parallel storage system is designed to run based on iRODS, the data grid software. The architecture of the IDA-based data parallel storage system is shown in Figure 2 as follows.

In principle, iRODS is designed as a virtual file system with a metadata catalogue and a file access API. The metadata catalogue, iCAT of iRODS, stores name-value-unit triples in a relational database and presents mechanisms to save, delete, or query metadata. The file access API offers a virtual file system for users to access remote data in a similar way as using local file systems. It is configurable to get access to data in various storage media. iRODS provides an infrastructure for data intensive applications in scientific research areas to fulfill massive data file storage across different distributed storage media. With client tools, such as iCommands and iRODS explore, researchers can manage large petabyte scale data collections. Moreover, the iRODS programmable interfaces, such as Jargon API, have been developed and provide an approach for users to develop additional custom functions and to integrate with their own software systems.

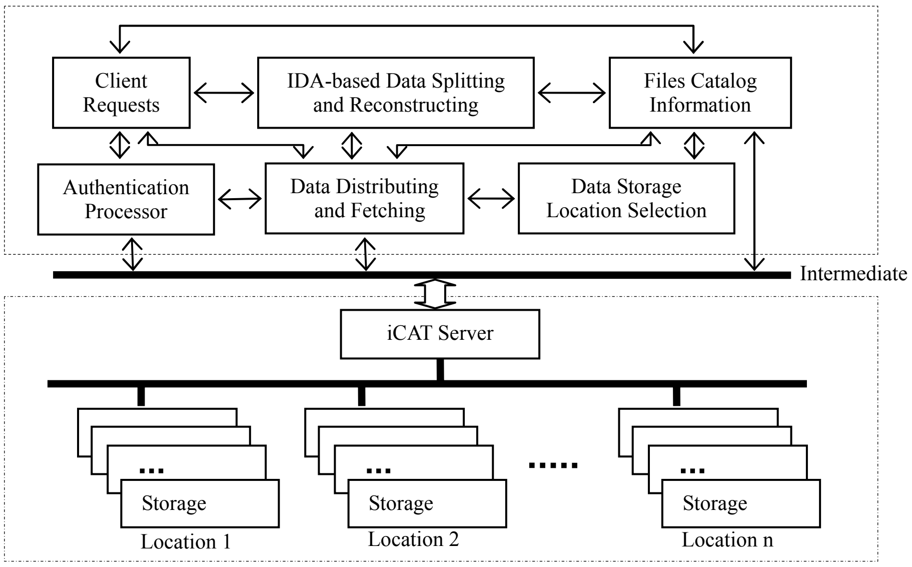

Client Requests
IDA-based Data Splitting and Reconstructing
Files Catalog Information
Authentication Processor
Data Distributing and Fetching
Data Storage Location Selection
Intermediate
iCAT Server
Storage
Location 1
Storage
Location 2
Storage
Location n

Figure 2. The data parallel storage system architecture

The IDA-based data parallel storage system implements the data file state querying, storing, and fetching functions based on the information dispersal algorithm and iRODS system. It consists of six modules: Client Requests, Authentication Processor, IDA-based Data Splitting and Reconstructing, Data Distributing and Fetching, File Catalog Information, and Data Storage Location Selection, as shown in Figure 2.

Among the six modules, the Client Request module is an interface to users and administrators. Through this interface, users perform data file storage information querying, data file storing to and fetching from the data grid. In order to ensure system security, the Authentication Processor module has been designed and is triggered when users start a data session and perform some operations. The IDA-based Data Splitting and Reconstructing module performs the information dispersal algorithm processing to divide input data file into block files and reconstruct block files fetching inversely from the data grid into the original file. The Data Distributing and Fetching module is the interface to the data grid system that handles file transmission to and from storage servers of the data grid. Similar to the data grid and distributed file systems, the whole data parallel storage system also needs an important module named Files Catalog Information to store all the system's metadata information. The Files Catalog Information module stores all system information, i.e. the original file information, blocks file information, IDA parameters ($n, m, A_{n\times m}$), and storage location information, etc.

It should be noted that the Data Storage Location Selection module is designed to rank and list the quality of the access performance of different storage locations according to a priority defined by users themselves and system conditions. Different storage locations may have distinct quality rankings, depending on three factors used to generate a final grading, from which we evaluate the quality of storage locations: network bandwidth, transmission distance, and history of access performance. Consequently, the data file can be distributed across storage locations with high data access performance and can be fetched efficiently in parallel.

When storing a file in the IDA-based data parallel storage system, the user logs into the system and data grid and then chooses preferred storage locations in the data grid and IDA parameters. The input file is next processed by the IDA-based data splitting and reconstructing component, which divides it into *n* block files distributed across the data grid by the data distributing and fetching module. Because the data distributing and fetching module can transport the block files to different storage servers in parallel and concurrently, high data transmission performance can be achieved. At the same time, the quality of each storage server is evaluated and stored for further reference by the data storage location selection module. Finally, information about the original file, block files, and storage locations is stored in the files catalog information module.

To download a file from the parallel storage system, the information about the file and *n* block files and storage locations is obtained from the files catalog information module. Then the *m* block files stored in the best storage locations with high rank are chosen by the data storage location selection module. The *m* block files are downloaded in parallel with high performance from the different storage servers. Eventually, the *m* block files are used to reconstruct the original file by the IDA-based data splitting and reconstructing module.

4 EXPERIMENTS AND ANALYSIS

4.1 Simulation Environment

In this section, we demonstrate a simulation environment to evaluate the data access efficiency and performance of the IDA-based data parallel storage strategy compared to replication technology. For simplicity, we implement all the modules of the IDA-based data parallel storage system on one single server and through the server access a small data grid subsystem in parallel based on the integrated rule-oriented data system. The simulation environment is shown in Figure 3.

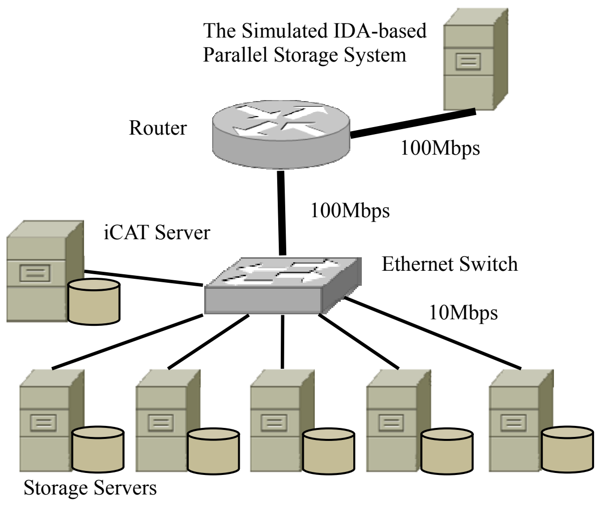


Figure 3. The environment of the simulated IDA-based data parallel storage system

In the simulation environment, the data grid subsystem consists of one metadata catalog server and five storage servers connected by an Ethernet switch configured with 10Mbps port bandwidth. The simulated IDA-based parallel storage server with a 100Mpbs network interface card connects to the data grid subsystem by a router with 100Mbps port bandwidth. The bandwidth between the switch and the router is configured to be 100Mbps.

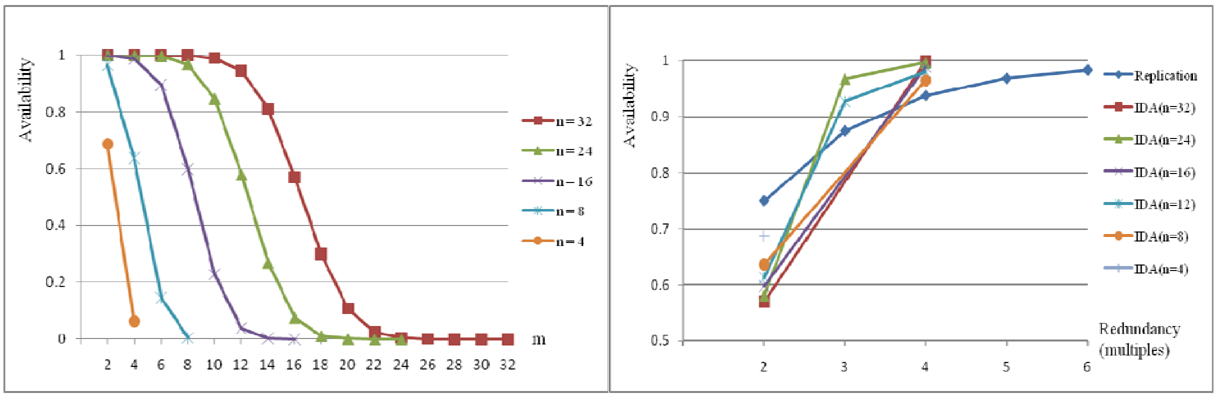
4.2 Data Availability Analysis

In this section, we give a detailed theoretical analysis of the data availability of the IDA-based parallel storage scheme compared with that of replication technology.

From the previously described system overview section, we know that the information dispersal algorithm tolerates up to *(n-m)* failures. Furthermore, we assume that all the involved storage servers have an independent probability of failure *p*. Therefore, the data availability of the IDA-based parallel storage scheme can be drawn as shown in Equation 9. The data availability of the replication technology is determined by $(1-p^k)$ as shown in Equation 10, which means that data access fails only when all the replicas break down.

$$Availabili\,ty_{IDA} = \sum_{i=0}^{n-m} \binom{n}{i} P^i (1-P)^{n-i} \tag{9}$$

$$Availability_{replication} = 1 - P^k \tag{10}$$



(a) Availability over *n* and *m*, where *p=0.5*

(b) Availability compare with replication, where *p=0.5*

Figure 4. Data availability of the IDA-based parallel storage scheme and the replication method

Suppose that the probability of failure *p* is 0.5. We can theoretically calculate the data availability of the IDA-based parallel storage scheme by Equation 9, when the parameter *n* of IDA is set to be 4, 8, 16, 24, and 32. From chart (a) of Figure 4 as shown above, data availability increases as the value of *n* increases under the conditions of a fixed value of *m* because only a small subset of blocks are needed to reconstruct the original data file. We can see that the data availability approaches the value 100% when the parameter *n* is much larger than *m*.

Chart (b) of Figure 4 shows the difference of the availability trends toward storage redundancy between the

IDA-based parallel storage scheme and the replication technology. Storage redundancy means the number of replicas for replication technology and the multiples of *n* and *m* for the IDA-based parallel storage scheme. From chart (b) of Figure 4, we can see that the IDA-based parallel storage scheme has better performance than the replication technology when storage redundancy is greater than 2. For example, when the redundancy is 3, the availability of the IDA-based parallel storage scheme (i.e. $n=24$, $m=8$, $p=0.5$) is 0.968 from Equation 9, whereas the availability of replication technology from Equation 10 is 0.875 under the same conditions (i.e. $k=3$, $p=0.5$). Furthermore, the availability of the IDA-based parallel storage scheme can far more quickly reach the value of 100% than can replication technology with the redundancy increasing.

4.3 Parallel Access Performance Analysis

In the following, we present a detailed analysis and develop a data file access experiment to compare the performance of the IDA-based parallel storage scheme and the replication technology. We evaluate the download time to measure performance, denoted T_{IDA} and $T_{Replication}$, and formalized, as shown in Equations 11 and 12, respectively. The parameters involved in the two equations are described in Table 1.

As is shown in Equation 11, the file download time of the IDA-based parallel storage scheme is composed of two parts: one is the block files download time; the other is the IDA processing time to reconstruct the data file. We generally assume there are *m* block files scattered on *m* storage servers needed to reconstruct the original data file. Therefore, the download time is determined by the longest transmission time among the *m* sessions because the *m* block files are downloaded concurrently. Moreover, the block files download time can also be divided into three portions: time for session establishment (Q_i), time for being read from disks (S_i/DB_i), and time for being transferred through the network (S_i/NB_i). Then the time for the IDA processing to reconstruct the original file (Q_{IDA}), which can be formalized in Equation 13, is dependent on the specific system performance on which the IDA is running. From Equation 13, it is apparent that this amount of time is in direct proportion to the value of data file size *L* and parameter *m* of IDA. In our simulation, this is much smaller than the total download time.

The time consumed for accessing a replicated file in a data grid environment is normally expressed as Equation 12. The minimum value among transmission times is determined by the downloading from *k* storage servers. Similarly, the file download time of replication technology includes three portions: time for session establishment (Q_i), time for being read from disks (L/DB_i), and time for being transferred through network (L/NB_i).

Table 1. The parameters of the IDA-based parallel storage scheme

Parameters	Description
L	The size of the data file
S_i	The size of the block files divided by the IDA
DB_i	the disk bandwidth of the *i*th storage server
NB_i	the network bandwidth of the *i*th storage server to access point
Q_i	the time consumption of TCP connection establishment
Q_{IDA}	the overhead of IDA processing to reconstruct the data file
r	equal to L/m denoting the length of each file block
m	the number of blocks required to reconstruct the data file

n	the block number of data file divided by the IDA
k	the number of replicas of a data file
IM	the integer multiplication instruction
IA	the integer addition instruction
T_{IDA}	the time consuming to download a file stored in IDA-based parallel storage system
$T_{Replication}$	the time consuming to download a file or its replica stored in a specific storage server

$$T_{IDA} = MAX_{i=1,2,\ldots,m}\left(Q_i + \frac{S_i}{DB_i} + \frac{S_i}{NB_i}\right) + Q_{IDA} \tag{11}$$

$$T_{\mathrm{Re}\ plication} = MIN_{i=1,2,\ldots,k}\left(Q_i + \frac{L}{DB_i} + \frac{L}{NB_i}\right) \tag{12}$$

$$Q_{IDA} = r*m*(m*IM + m*IA) = L*m*(IM + IA) \Rightarrow O(L*m) \tag{13}$$

We have conducted a performance simulation to compare the performance of the IDA-based parallel storage scheme with that of replication technology. The simulation environment is shown in Figure 4, and the results are shown in Figure 5. In the simulation, data file sizes of 40MB, 80MB, 120MB, and 160MB are selected and scattered in five storage servers for the download performance test. The parameters *m* and *n* of the IDA are set to be 4 and 8 respectively, so there are four block files to be downloaded concurrently from four storage servers to reconstruct the original data file. As to the replication technology performance test, the performance result is acquired by downloading the data file from one storage server for simplicity. The resulting values of total download time presented are the average of several test results.

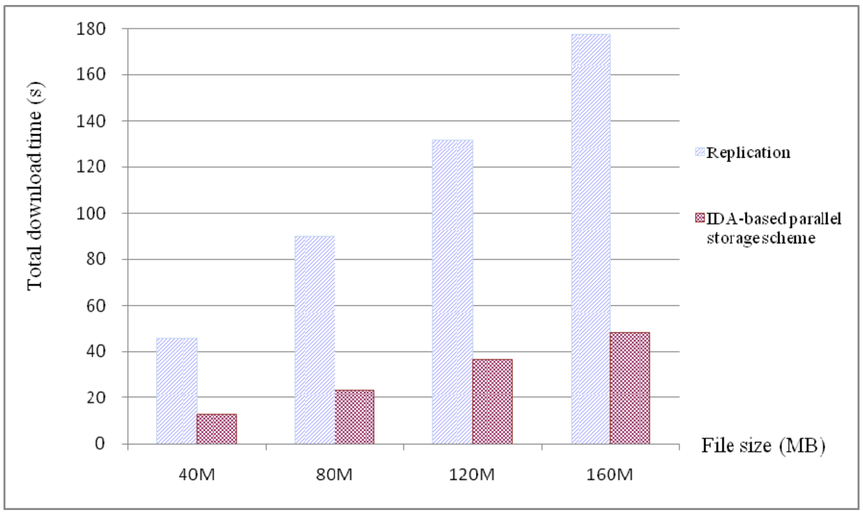


Figure 5. Comparison of IDA-based and replication methods download times

From the results shown in Figure 5, the IDA-based parallel storage scheme presents a much better performance than replication technology. Note that each session between a storage server and the access point (the simulated IDA-based parallel storage system) has 10Mbps bandwidth. Therefore, the aggregation bandwidth of the session in the IDA-based parallel storage scheme test is four times as much as the one in the replication technology test.

As expected, the file download time of replication technology is more than three times as much as that of the IDA-based parallel storage scheme because of concurrent file transmission and the overhead of the IDA file processing. Furthermore, as the size of the data file increases, the time consumed in the IDA processing to reconstruct the original file rises accordingly.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we investigate several data storage and data access strategies in detail and propose a new IDA-based parallel storage scheme in the data grid environment. Replication technology takes much more space and achieves a smaller amount of data availability than the IDA-based parallel storage scheme. Moreover, the IDA-based parallel storage scheme makes full use of data parallel transmission technology by accessing the geographically scattered block data files in different locations and achieves higher data access performance than replication technology. The theoretical analysis and simulation experiment confirms that the IDA-based parallel storage scheme improves both data availability and data access performance in a data grid environment. Furthermore, by virtue of the high data availability and the smaller storage space occupation, this scheme has the potential to reduce considerably storage requirements for large-scale databases on a data grid.

In the future, we plan to work on improving the performance of the information dispersal algorithm to process large data files of gigabytes efficiently. Additional attention will be paid to improving data transmission performance, adopting multithreads dynamically within each session between storage server and access point. We will try to implement a parallel storage system to run on separate servers to improve performance. Ultimately, we plan to improve the prototype of the IDA-parallel storage scheme, not only to verify the effectiveness as a proof-of-concept but also to try to instantiate it in a working data grid environment.

6 ACKNOWLEDGEMENT

This research is supported by the Special Project of Informatization of Chinese Academy of Sciences in "the Eleventh Five-Year Plan" under grant NO. INFO-115-C01 and INFO-115-D02.

7 REFERENCES

Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., & Foster, I. (2005) The Globus Striped GridFTP Framework and Server. *Proceedings of the 2005 ACM/IEEE conference on Supercomputing, ACM Press*.

Baru, C., Moore, R., Rajasekar, A., & Wan, M. (1998) The SDSC Storage Resource Broker. *Proceedings of the 1998 Conference of the Centre for Advanced Studies on Collaborative Research.*

Brinkmann, A. & Effert, S. (2007) Cost-effectiveness of Storage Grids and Storage Clusters. *The 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing*.

Changa, R., Guob, M. H., & Lin, H-C (2008) A multiple parallel download scheme with server throughput and client bandwidth considerations for data grids. *Future Generation Computer Systems 24*, 798–805.

Data Intensive Cyber Environments Group, University of North Carolina at Chapel Hill, University of California at San Diego (2008) iRODS: integrated Rule Oriented Data System White Paper. Retrieved October 15, 2009 from the WWW: http://www.irods.org/

Davis, S. (2008) Progress Towards Efficient Data Ingestion into iRODS. Retrieved October 21, 2009 from the WWW: http://www.irods.org/

Dean, J., & Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM archive 51*(1).

García-Carballeira, F., Carretero, J., Calderón, A., García, J. & Sanchez, L. (2007) A global and parallel file system for grids. *Future Generation Computer Systems 23,* 116-122.

Ghemawat, S., Gobioff, H., & Leung, S-T. (2003) The Google File System. *ACM SIGOPS Operating Systems Review 37*(5).

GT4: Globus Toolkit 4. Retrieved October 16, 2009 from the WWW: http://www.globus.org/

Gu, YH. & Grossman, R. (2008) Exploring Data Parallelism and Locality in Wide Area Networks. *Many-Task Computing on Grids and Supercomputers*.

iRODS. Retrieved October 15, 2009 from the WWW: http://www.irods.org/

Li, K-C., Wang, H-H., Cheng, K-Y., & Wu, T-Y. (2009) Strategies Toward Optimal Access to File Replicas in Data Grid Environments. *Journal of Information Science and Engineering 25*.

Moore, R. W. (2007) Managing Large Distributed Data Sets Using the Storage Resource Broker. *ITEA Journal*.

Rabin, M. O. (1989) Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM 36(2),* 335-348.

Raicu, I., Zhao, Y., Foster, I., & Szalay, A. (2008) Accelerating Large-Scale Data Exploration through Data Diffusion. *Proceedings of the 2008 international workshop on Data-aware distributed computing.*

Rodriguez, P. & Biersack, E. W. (2002) Dynamic Parallel Access to Replicated Content in the Internet. *IEEE/ACM Transactions on Networking 10*(4).

Secretan, J., Lawson, M., & Bölöni, L. (2009) Efficient allocation and composition of distributed storage. *The Journal of Supercomputing 47*(3), 286-310.

Wang, C. M., Hsu, CC. & Wu, JJ. (2007) A High-Performance Virtual Storage System for Taiwan UniGrid. *Journal of Information Technology and Applications 1*(4).

Wylie, J. J., Bakkaloglu, M., Pandurangan, V., Bigrigg, M. W., Oguz, S., Tew, K., Williams, C., Ganger, G. R., & Khosla, P. K. (2001) Selecting the Right Data Distribution Scheme for a Survivable Storage System. *Technical Report CMU-CS-01-120*, Carnegie Mellon University.

(Article history: Received 2 February 2010, Accepted 7 May 2010, Available online 19 May 2010)