Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 2
May 2009

SDS: A Scalable Data Services System in Data Grid

Xiaoning Peng
School of Information Science & Engineering, Central South University
Changsha 410083, China
Department of Computer Science and Technology, Huaihua College
Huaihua 418008, China

Bin Huang
Department of Computer Science and Technology, Huaihua College
Huaihua 418008, China

Ping Luo
Department of Computer Science and Technology, Tsinghua University
Beijing 100084, China

Abstract

It is very complex and of low efficiency for grid users to access heterogeneous diverse data resources distributed over the whole wide area network. Data Grid has denoted a network of storage resources, from archival systems, to caches, to databases, that are linked across a distributed network. And it should provide integrated, scalable data services which implement more wide range of transparent access to data resources in Data Grid, such as location transparency, time transparency. In this paper, we describe a SDS system which can implement integrated, scalable data services. We have implemented one of the important building blocks for SDS is the server called DSB which can provide integrated data services. The DSB bases on cluster and agent technologies. Agent-based DSB can cleverly prefetch required data, replicate them among DSBs. Each DSB based on cluster implements a single entry and a virtual integrated storage system in a data domain. As far as whole SDS, Multiple DSBs are formed cluster data services, which are scalable, and provide a single entry for all data grid users, and provide a virtual integrated mass storage system, and hide distributed heterogeneous low-level data resources, and insure load balance of each server. SDS architecture supporting these various scenarios, are also described.

Keywords: Data grid, Data services, Scalable, SDS system, DSB

1. Introduction

An increasing number of applications in domains such as genomics/proteomics, astrophysics, geophysics, computational neuroscience, or volume rendering, need to archive, retrieve, and process increasing large datasets(Keith Bell, Andrew Chien, Mario Lauria). These data-intensive applications are prime candidates for Data Grid as they involve remote access and extensive computation to many data repositories. Data Grid has come to denote a network of storage resources, from archival systems, to caches, to databases, that are linked across a distributed network. One of the core problems that any Data Grid project has to address is the heterogeneity of storage systems where data are stored. These can be either mass storage management systems like HPSS, Castor, UniTree, and Enstore, multiple disk storage systems like DPSS, distributed file systems like AFS, NFS(R. Sandberg, 1987), or even databases. This diversity is made explicit in terms of how data sets are named and accessed in all these different systems(Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger, 2000). For instance, in some cases data are identified through a file name whereas other systems use catalogues where data are identified and selected by iterating over a collection of attributes or by using an object identifier.

So data services which implement uniform access to distributed data sources are the important research part and the key technology of Data Grid. We believe that a wide range of transparencies is important for data services in Data Grid.

One of the important building blocks for any Data Grid is the server implementing data services(Keith Bell, Andrew

Chien, Mario Lauria), so in this paper, focusing on the server, SDS adopt some approaches to implement more wide range of transparent data services in which the following strategies are used to address the purpose:

- We assume a cluster-based architecture for our server because it uses inexpensive off-the-shelf PC components, offers an inherently scalable aggregate I/O bandwidth, well adaptability of tape and network, dynamic task scheduling, etc. By leveraging the high-speed communication afforded by the cluster, large files can be stored in a scalable fashion by striping the data across multiple nodes(Keith Bell, Andrew Chien, Mario Lauria).
- We adopt agent technologies to provide data prefetching mechanisms, which implement available data are replicated to Client without application fetching them from remote storage. Agent-based SDS has many advantages such as autonomy, responsive, proactive, objective-oriented, etc.

In the rest of this article, we first describe in Section 2 the architecture of SDS and DSB (Data Services Broker), Then in Sections 3, 4, 5 describe the design of cluster DSB, respectively. We discuss intelligent prefecthing mechanisms in Section 6. Finally, conclude in Section 7.

2. SDS Architecture

Data Grid aggregates many diverse storage systems geographically distributed over wide area networks, abstracts these storage system as a single virtual data object, shares these data resources located in different administrative domains. Data grid should have many DSBs distributed in independent administrative domains. Every DSB manage all data resources located in these domains. All DSBs will be federated to provide integrated data services for users or applications. In SDS, DSB substitute for users to access storage resources and finally complete all users' requests.

We design SDS that is a typical distributed system adopting distributed design patterns. SDS includes three tiers: client, DSB and meta-server, storage resources. DSB addresses to access and management of the grid data and other resources. It's the core of SDS. It aggregate organically various modules and works orderly, and support the attribute-based access to data collection and items and other system resources. SDS employs many distributed meta-server to enable the metadata services to have high scalability. These meta-servers can be classified into two categories: Local Meta-Server and Global Meta-Server. Local Meta-Server store the local metadata which are associated with the data located in local domain; Global Mata-Server store the global metadata which are the index of the whole metadata. Therefore, when a DSB failed to find the required metadata in certain Local Meta-Server, it can get the metadata index and finally achieve it. The topology of DSB and meta-server is shown in Figure 1.

The deployment of DSB is based on site. Each site has a DSB server providing data access service. High-speed transport protocol such as Gridftp, is used to transfer data between DSB and Client. DSBs communicate with each other and provide together federated services for clients. Meta-Servers are independent of DSBs. Their relationships are created by configuration. It may not be deployed based on site. Each site can have one meta-server, or several sites have one, or the whole system has one meta-server.

The scheduler of request replaced before all DSBs parses access requests and routes clients' requests to proper DSB.

3. SDS Design and Implementation

Adopting three-layer architecture, we designed SDS, whose main components include Client tools, DSB (Data Services Broker) servers and metadata servers (MDIS), as illustrated in Figure 2.

The first layer is Storage Environment (SE) which consists of all kinds of physical storage resources and metadata resources, including all kinds of file systems, archive systems and database systems. It accesses and operates the datasets in these systems through native protocols and methods that the physical resources support. Metadata resources include data metadata, replica metadata, user metadata, access control metadata, system metadata, configuration metadata and application metadata, etc.

The second layer, i.e. the service layer, is the core of the system. In the layer, we abstracts the storage system distributed over the whole wide area networks as a single virtual data object, and define common operations on the virtual data object. The major components of the layer is a server which is considered as Data Services Broker (DSB), because through it grid users can access all data services provided by SDS and eventually users' requests are satisfied.

DSB is designed in the light of master-slaver pattern. In DSB, slaver program called Proxy composes of *Access scheduler*, *Replica Manager*, *GridFTP Controller*, many *Accessor*, and so on.

Access scheduler which is the center of Proxy controls the order of calling other module.

Replica Manager moves and copies data around the world either according to predefined policies or on demand of user or broker. The *Replica Manager* keeps track of such movements through the Replica Catalogue. The *Replica Manager* can do one or more of the following:

- Periodically verify the contents of an RC by checking with the SE on the existence and status of files that should be contained therein.

- Schedule *Accessor to* Copy one or more files from a source SE to cache in DSB according to access protocol and storage mechanism of objective data, then call *GridFTP Controler* to transfer data to a target SE or client using GridFTP.
- Estimate the elapsed time required to perform a replication. For this it requires information of the size of a set of files, the time to service the requests, network bandwidth figures between the source and target SEs.
- Perform a more sophisticated estimate of the time taken to access different replicas of the same file in order to decide which replicas to use.

Accessor is responsible for all aspects of data access from underlying repositories. DSB has many Accessors. An *Accessor* is in charge of accessing to a kind of storage system.

GridFTP Controller control data transfer between client-side cache and server-side cache by Controlling GridFTP(A. L. Chervenak, I. Foster, C. Kesselman *et al*. 2002) Servers running on the client and server.

Security Service provide some mechanisms for security include authentication, authorization, prevention from attack and access control of the data object distributed over wide area networks and different autonomous domains. DSB also offer a single-sign on mechanism for identifying the grid user.

Monitoring Services provide real-time and accurate local and global information of grid for upper levels to ensure good running of grid storage system. It must provide functions such as monitoring and early-warning of resources' performance, alarming and real-time processing on resources failing, monitoring and early-warning of software running, recovery of software failure, etc.

Data publishing service is the service that a data source could explicitly publish its schema and content to one or more metadata servers if it believes that the data is of value to many users, and can also publish its capabilities for querying the content.

DSB hides from higher layers the complexities and specific mechanisms for data access which are particular to every storage system manipulating the performance factors that are proprietary for each system. Using the core services, DSB provide more transparencies for users and applications to manage multiple data sources and access data.

Cache Manager has the primary role of a *disk pool manager* whose functions are to allocate space on disk to store a file, to maintain a catalogue (or list) of all the files in its cache and to clear out (delete) old or least recently used files in these cache when more free space is required, which is a process known as garbage collection.

The third layer is the interfaces called portal that our system provides for users and applications. Portal provides a single entry of all kinds of grid data resources, including uniform data access interface, uniform data and system management interface, data ordering interface, workflow and dataflow customizing interface for users. It combines the application, data and workflow to make a integrated application environment that support data access, transport and management across diverse administrative domains. There is a client cache pool and its management tools, GridFTP Server, configure tools in the layer too. Client cache management tools have the same functions as Cache Manager in DSB. Configure tools setup some parameters to enable client-side programs conveniently run.

Two layers cache mechanism is introduced to SDS. DSBs replicates a lot of data to server-side cache in order to prepare some data for clients. Clients access to data from client-cache using native access mechanisms. DSBs utilize GridFTP to exchange data between Client and DSB.

4. Cluster-Based DSB

In order to improve the performance of Data Services, We assume a cluster-based architecture for DSB because it uses inexpensive off-the-shelf PC components, offer an inherently scalable aggregate I/O bandwidth, and can take advantage of existing cluster installations through double-use or upgrade of older hardware. By leveraging the high-speed communication afforded by the cluster, large files can be stored in a scalable fashion by striping the data across multiple nodes.

Each DSB is designed by adopting cluster-structure. In cluster servers, there are two types of nodes: master node and slave node. Service program also has two parts: Manager, Proxy. Manager running on master node, receive users' requests and maintain the global status of its DSB. When receiving a user's request, it creates a proxy process in a proper slave node. Other operations will be completed by that Porxy interact with Client hereafter. Manager can know the status of each slaver node and accomplishes proper load balancing and task scheduling mechanisms.

Proxy running on a slaver node achieves data replication, reading and writing according to the storage mechanism of objective data. Proxy provides uniform convenient data access interface for users to acquire platform-independent data with high efficiency. Each Proxy is independent of manager. When user send connection request to DSB, Manager chooses the best node according to some policies and start an Proxy process on it. Then Manager returns the IP and port of the slave node to the client. All communications, instructions and operations are accomplished by interaction

between Proxy and clients. The Architecture of cluster-Based DSB is illustrated in Figure 3.

Slave node not only maintain running environment of an Proxy, but also has the function of data cache. The data prefetched by Proxy and real-time data are all cached into slave node. Slave node starts high-speed transport tool to transfer the data to client's cache. Then client can operate the data object using local access mechanisms.

5. Federated Data Services

This combines two aspects: all DSBs form a cluster, Multi-domain's DSB collaborative data access.

Multiple DSB servers can collaborate to form a federated DSB which acts like one single server and provides uniform services for data requests. As illustrated in Figure 3, the request scheduler which is placed before all DSBs implement a single entry point, and provides a virtual integrated storage system for users and hides the back DSBs' distributed property. The request scheduler also monitors DSB and its load balancing for it is the only entry of the cluster. Tasks on each DSB are dispatched by request scheduler. So it can know the status of each node inside SDS and accomplishes proper load balancing mechanism to make SDS has very high performance.

There are many DSBs in our system and each DSB manages a local domain. So when users access a data object which isn't located in local DSB, the local DSB needs to communicate with another DSB in which the data object is located. The two DSBs are to complete the task together. Figure 4 shows this access process.

From Figure 4, we can see that when client sends data request to master node in server A, the master node in server A creates a Proxy process in slave node to access meta-server. If the Proxy cannot find corresponding data object in local Meta-Server, then local Meta-Server queries the global meta-server and return the address of DSB in which the data object located (for example B) to the Proxy. Then the Proxy sends request to server B and the master node in server B creates a Proxy serving for it in a certain slaver node in the same way. All the instructions are exchanged between the two Proxies. But the data are directly transferred to client using Grid FTP by the Proxy of server B.

6. Intelligent Data services

An agent is a computer program that monitors some environment and performs actions to modify it as conditions change, and that is capable of autonomous action in this environment. We adopt the agent technology to implement intelligent DSB because the data services based on agent technology has many advantages such as autonomy, responsive, proactive, objective-oriented, etc. The intelligent data services combine two aspects:

- Cleverly prefetching data

Agent-based Proxy which was discussed in section 4 is an intelligent program entity based on agent technologies running continually and autonomously in a specific environment. It can prefetch, cache data basis on some policies, such as the history of users' data access, data section principle, etc; redirect users' visits and other operations to other proxy in which the required data have been cached. Each Proxy in a DSB can communicate with each other, collaborate and have learning abilities, so some optimized policies can be taken to improve the data access efficiency during the information acquiring.

- Cleverly replicating data and dispatching task among DSBs

Each DSB is also intelligent and can learn from each other. By learning, more free DSBs can know the list of data which more busy DSBs often access and replicate these data. It can redirect users' visits and other operations to other DSB according to the distribution and frequency of data visited by users. DSBs that are busier can know the DSB whose load is the lightest by communication and then redirect client's access request to it.

7. Conclusion

In this paper, we have described SDS and emphasized DSB to improve its performance. DSB utilized two technologies: cluster, agent. SDS has the following advantages:

(1) DSB abstracts the storage system distributed over the whole wide area networks as a virtual integrated data object, and hides their distributed diverse property, provides integrated data services. The Request Scheduler hides all DSBs' distributed property, achieves a single entry.

(2) The slaver node in every cluster DSB can cache a lot of data, besides communicating with clients and processing the request of clients. We can see that the capacity of each DSB's cache (C_{total}) is proportional to the number of slave nodes(N) in the cluster DSB composed of homogenous slave nodes($C_{total} = N*C_s$, C_s is the capacity of a DSB's cache, and it is same for all homogenous slave nodes). Therefore this ensure that the capacity of each cluster DSB(C_{total}) is scalable.

(3) Similarly, we can see that when the network bandwidth is enough, the throughput of each cluster DSB is scalable. Because the throughput of each DSB (R_{total}) is proportional to the number of slave nodes (N) in a cluster DSB composed of homogenous slave nodes, i.e. $R_{total} = N*R_s$ (R_s is the throughput of a slaver node). So we can implement

network–striped transport and achieve aggregate cluster-to-cluster throughput which scales with the number of connections. This enables our system to suit the cluster-to-cluster high performance computing environment.

(4) Two layer caches adopted in SDS hide the data transfer time, effectively decrease the data access response time, and implement the time transparency of the data services. Furthermore, the large capacity of data cache introduced in our cluster DSB decreases miss rate.

(5) Agent technologies make our system is intelligent. So, DSB can cleverly prefetch the required data and move it to client in time; Data can be cleverly replicated among DSBs; Tasks can be cleverly redirected among DSBs.

(6) Whether single DSB or SDS are implemented according to cluster structure, This can achieve load balance and insure the high availability of whole system for cluster have good failure tolerance.

(7) SDS is introduced multi-layers meta-server and based cluster, so it is scalable, and can be flexibly deployed and configured.

References

A. L. Chervenak, I. Foster, C. Kesselman *et al*. (2002). Data Management and Transfer in High Performance Computational Grid Environments. Parallel Computing Journal, 2002, 28 (5): 749-771.

DPSS: Distributed Parallel Storage System, http://www-itg.lbl.gov/DPSS/

Enstore: http://www-isd.fnal.gov/enstore/design.html

Keith Bell, Andrew Chien, Mario Lauria, *A High-Performance Cluster Storage Server,* 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)

R. Sandberg. (1987). The Sun Network File System: Design, Implementation and Experience,Tech. Report, Mountain View CA: Sun Microsystems, 1987.

UniTree: http://www.unitree.com/

Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. (2000). Data Management in an International Data Grid Project, http://www.eu-datagrid.org/, 2000

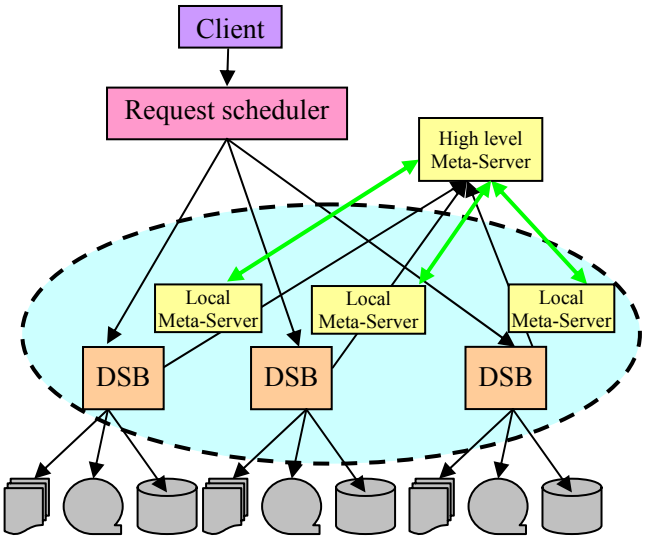

Figure 1. The Architecture of DSB

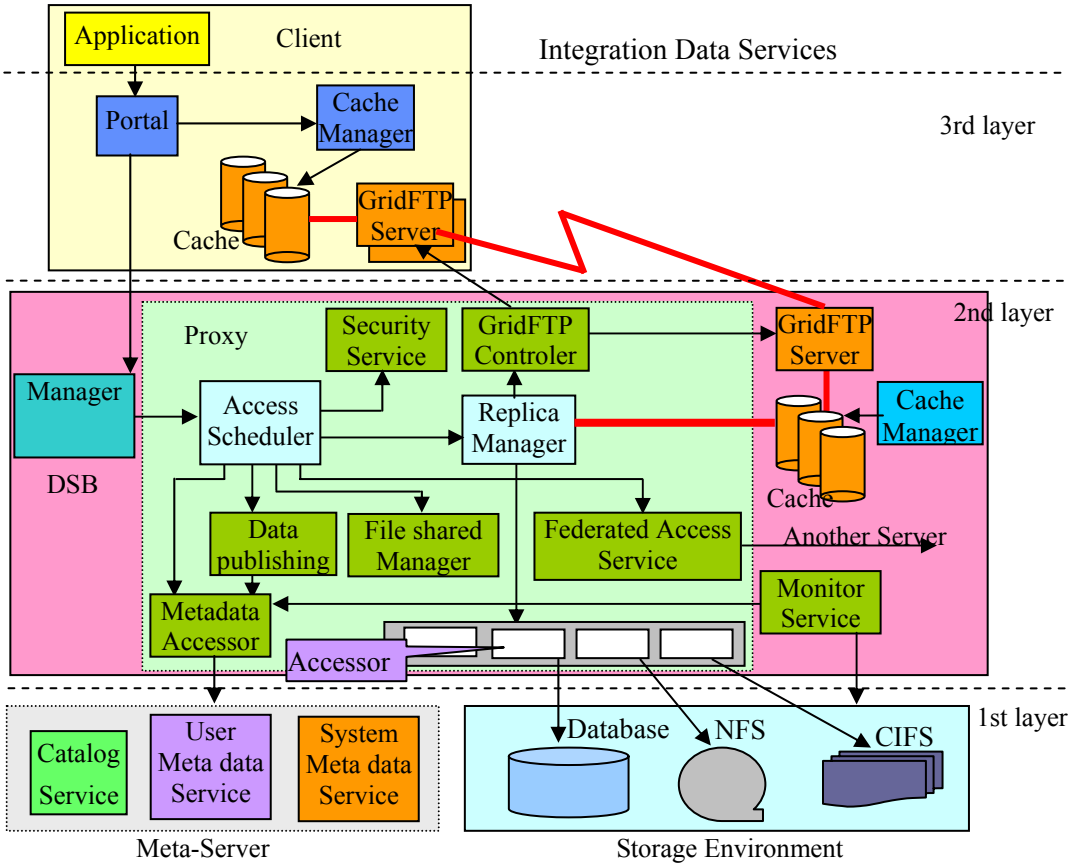

Figure 2. The Components of SDS

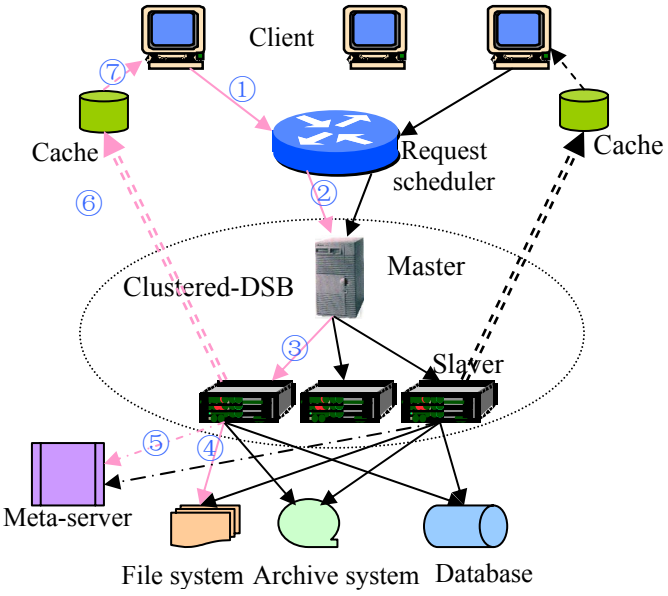

Figure 3. Cluster-Based DSB

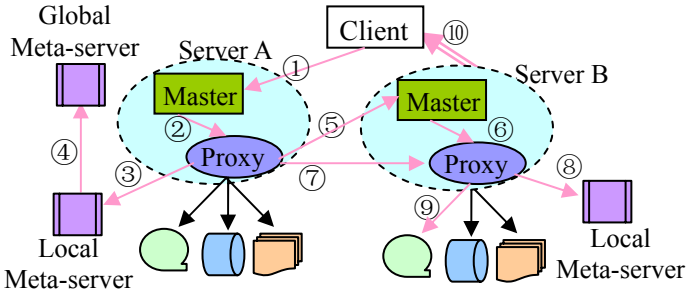

Figure 4. Two DSB Federated Services