RESEARCH AND BUILD DATA GRID FOR GOODAS SYSTEM

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF SCHOOL OF
INFORMATION AND COMMUNICATION TECHNOLOGY
AND THE COMMITTEE ON UNDERGRADUATE STUDIES
OF HANOI UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF OF ENGINEERING

To Trong Hien
June 2010

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Nguyen Thanh Thuy) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Nguyen Huu Duc
(Research Advisor))

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Nguyen Hong Thanh
(Co-worker))

Approved for the University Committee on Undergraduate Studies.

Preface

Writing this thesis has been hard but in the process of writing I feel I have learned a lot..

Abstract

Currently, the demand for searching and document sharing to serve for research and teaching at university is so enormous. Besides, cheating situations like copying the thesis from previous semesters or even from other universities are happening sophisticatedly and popularly. This negatively affected to the quality of the thesis particularly and the quality of education in general. Therefore, the need for a document sharing system which provide the ability for comparison in anti-fraud studying is very urgent.

However, managing distributed documents between universities is very challenge. They are dynamic resources which are belonging to different organizations and each organization has its own private policy in resource and access management. On the other hand, a distributed document managing system requires flexibility (easily add and remove storage and computing resource), security, and collaboration between research groups. To handle this requirement on a set of dynamic resources is complicated.

Develop from the demand; we propose and deploy a model system of electronic document management based on data grid technology GOODAS - Grid Oriented Online Document Analysing System. The system will connect computing resources and storage of dispersed universities for creating a grid inter-university data. On that basis, the system supports managing users, as well as searching document within the field or on the entire system. Especially, the system also provides the ability to match the document content, based on LDA algorithm - an approach to the semantics of analyzing the document content. For each input document, the system will rely on existing data warehouse to analyze the match for their content, and conclude how much the document is copied from other documents. Besides, we also conducted experiments grouped according to the document fields. Results showed that the system has enhanced the capacity to meet as well as the accuracy of the search and matching documents.

My job in the group is building data grid component which manages thesis according to research groups. This component must ensure these propersites of the system: security, reliability, availability and high scalability. The system (GOODAS project) is currently deployed for testing at High Performance Computing Center, Hanoi University of Technology.

Acknowledgements

I would like to thank our supervisor, Nguyen Thanh Thuy - Director of High Performance Computing Center, Hanoi University of Technology, for lots of great inspiration, ideas, comments of articles. Especially The Center is an ideal place where I have a chance to work with many senior researchers in parallel computing and grid computing.

Thanks to Dr. Nguyen Huu Duc, Ms. Nguyen Duc Tung and Ms. Dao Quang Minh , all from HPCC - HUT, for great support about orientation and advices as well as technique problems.

Thanks also to friends in AGC (Advanced Grid Computing): Nguyen Hong Thanh, Nguyen Viet Phuong and Nguyen Duy Hoang, all from HPCC research student group.

Finally, lots of thanks are due to my parents, my older and younger sisters for always support so that I can complete the thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to GOODAS system

1.1.1 GOODAS in general

Currently, the demand for searching and document sharing to serve for research and teaching at university is enormous. Besides, fraud situations like copy the thesis from previous semesters or even from other universities are happening sophisticatedly and popularly. This negatively affected to the quality of the thesis particularly and the quality of education in general. Therefore, the need for a document sharing system which provide the ability for comparison in anti-fraud studying is very urgent.

However, managing distributed documents between universities is very challenging. Firstly, the number of thesis is huge. Figure 1.1 shows the incremental rate of number of university and number of graduated student in Vietnam. From 2000 to 2008, the number of university increased from 178 to 393, and the number of student correlatively increased from 162.000 to 250.000 each year. This means that the volume of thesis increseas incessantly each year. Secondly, the documents are dynamic resources which belong to different organizations, and each organization has its own private policy in resource and access management. Thirdly, current distributed technologies have limitations. For example, CORBA and Enterprise Java are used

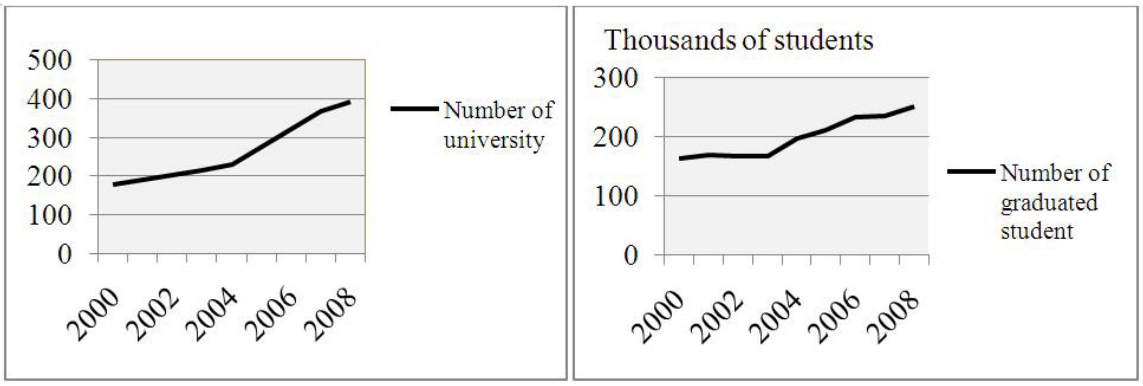

Figure 1.1: Increment of number of university and graduated student each year in Vietnam

popularly in distributed system; however, they are often used only in one organization, therefore, not suitable for an interscholastic system. Fourth, the students fraud situations are more sophisticated. For instance, they change the syntax and grammar, and use synonyms etc; hence, matching documents methods based on keywords does not meet requirements.

In recent years, grid computing in general and data grid in specific have incessantly advance. Grid computing technology allows combining computing power of many individual computers to create an enormous computing power. In this paper, we present a solution for managing distributed documents between universities. In addition, we deploy utilities about searching and matching documents approach towards semantics. Document-matching utility helps teachers in detecting plagiarism and academic fraud.

We propose a model of electronic document management based on data grid technology. The system will connect computing resources and storage resources of dispersed universities to create inter-university data grid. On that basis, the system supports managing users, as well as searching document within the field or on the entire system. Especially, the system also provides the ability to match the document content, based on LDA algorithm - an approach to the semantics of analyzing the document content. For each input document, the system will rely on existing data warehouse to analyze the match for their content, and conclude how much the document is copied from other documents. Besides, we also conducted experiments grouped according to the document fields. Results showed that the system has enhanced the respond time as well as the accuracy of searching and matching documents.

My job in the group is building data grid component which manages thesis according to research groups. This component must ensure these propersites of the system: security, reliability, availability and high scalability. The GOODAS project is currently deployed for testing at High Performance Computing Center, Hanoi University of Technology.

1.1.2 GOODAS's architecture and position of data grid

Our solution is connecting computing and storage resources of universities to create a common data grid, in which each university is a node and searching and comparing documents are executed simultaneously on each node. Then, we build grid portals for each university allowing users to access and use the system's functions. Grid portal provides access availability to the system's utilities to community through web; therefore, it makes grid transparent to the users. Figure 1.2 illustrates architecture model of GOODAS system. The model has four layers:

- Resource Layer: includes all basic resources, such as universities computing and storage resources as well as network infrastructure which linking distributed resources physically.

- Middleware Layer: includes many core services such as GridFTP (File Transfer Service), RLS (Replica Location Service), MDS (Monitoring & Discovery Service), and Credential Management. This layer allow developer to deploy grid-based applications. Nowaday, Globus Toolkit is one of the most common used grid middleware in grid computing projects in the world.

- Application Service Layer: provide application-based grid service and user-based grid service in order to utilize grid infrastructures power. On the scale of electric document managing system, this layer is deployed some utility services including discovery service, data service, searching service, and compare service.

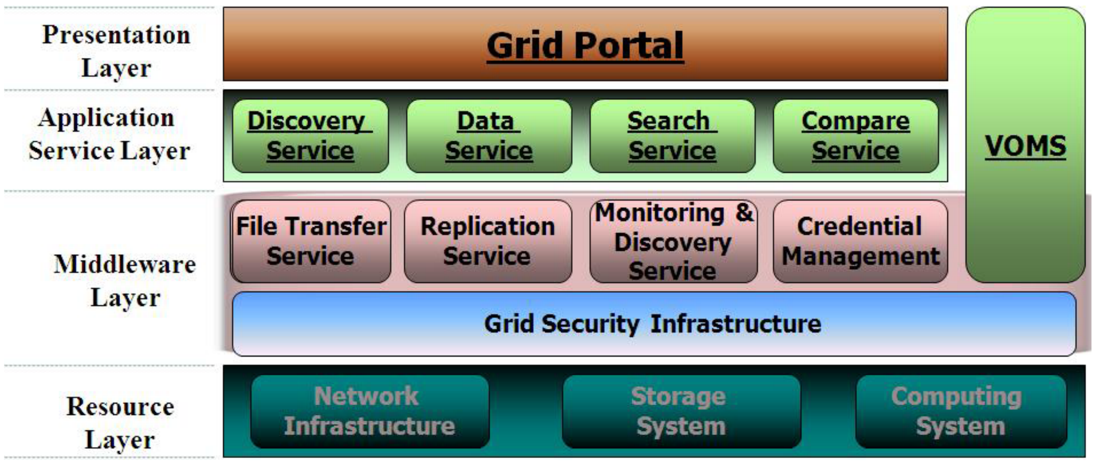

Figure 1.2: Architecture model of GOODAS system

- Presentation Layer: we use grid portal technology to connect users to grid system. The portal provides an unique common web to all computing and storage resources as well as utility services of the system. We can say that presentation layer make the grid transparent to users.

Users access GOODAS system through their university's portal, from there the users use services on local site such as searching and matching documents. For each user's request, the local site contacts the Information Server to use monitoring and discovery service to get information about other sites on the system and simultaneously send requests to process (as searching or matching documents) to the sites. After receiving results from other sites, the local site aggregates the results and return it to the users through portal. In order to synthetize the results, the local site should have metadata from other sites which are provided by Metadata Catalog Service on Information Server.

The position of the thesis on Figure 1.2 is setuping middleware layer, researching and deploying two application services: resouce discovery service and data service (replica service in specific). The next is deploying searching and matching document services.

1.2 Approach to data grid

The documents are stored in the distributed system over on multiple universities, so that issue is to build a data service layer based on data grid. This storage service connects resources from the universities to create a common storage layer which is transparent to the users, thereby provide utilities on data grid such as searching, matching document as well as download and upload...

Data storage services have already developed long time in the world, since Internet was popular with users. To build a strong storage service, we need suitable platform with appropriate technology. This technology platform must be able to combine multiple computer's powers. Through researching period, I realized that data grid technology is suitable for such the system. Data grid has the following advantages:

The ability to combine the organizations participating in the system: in term of the infrastructure of the university is still low while the typical storage system and data sharing requires a large amount of hardware resources. In order to overcome this disadvantage, data grid technology connects the resources of the participating university while ensuring the automomy of each university.

The stability: the architectures of data grid was used in many large scientific project worldwide such as LHC, caGrid...have shown the applicability of data grid is enormous. Grid computing technology can combine computers with different architectures, so that the change and upgrade of each node does not affect system in the whole. Each component in the grid is independent together, therefore, create the system for the system and avoid system the bottleneck problem of the whole system.

The Ability to create replication: the grid provide a powerful feature for the storage systems is creating and managing replicas. Replicas are made in order to increase reliablity and access performance. In the case of data in a particular storage node is not accessible, users can access the replicas of the data and continue to do the job without interuption.

1.3 The objective

The thesis 's objective is study data grid in general, then building GOODAS system's data grid component to manage document according to it's field group (computer science, mathematics...). The data grid should be able to create replicas of each group or all the documents between universities. The replication must ensure the autonomy of each university. My work is listed as follows:

- Research about grid computing and data grid to build an interscholastic data grid.
- Install middleware Globus Toolkit 4 [13], config storage computer to serve as data storage nodes.
- Create interfaces to access database and grid services to connect storage nodes to form a single data grid.
- Create, locate and manage interscholaslic replicas.

1.4 The content

Chapter 1: Introduction. This part introduce an overview of an interscholastic system for searching and matching document. The next is the necessity of a data storage layer for managing documents as well as an approach and the objectives of the thesis.

Chapter 2: Introduction to grid computing and data grid. This chapter presents general knowledge about grid computing and data grid particularly as features and architecture of data grid by Ian Foster - the father of grid computing. Later the chapter focus on introducing services that will be used for data grid like storage systems, file transfer service, replication services, information services...Last the chapter presents a tool for grid programming - grid service [14].

Chapter 3: Construction of data grid for GOODAS system. This chapter goes from analysis to design. The analysis part includes problem description, analysing functions and interaction scenario of the system. The design step includes overall design by presenting the model and detail designs such as database design, package and class design.

Chapter 4: Implementation and evaluation. This chapter presents deployment model and implementation results, then qualitative evaluations of the the data grid as well GOODAS systems.

Chapter 5: Conclusion and future development.

Chapter 2

Introduction to grid computing and data grid

2.1 Grid computing

In recent years, grid computing has emerged as a framework for supporting distributed computing and complex compilations over large data sets. The thought of the grid is to combine the computational power of thousands of normal computers geographically dispersed, creating a huge computing infrastructure equivalent or even beyond the power of expensive modern supercomputers.

Michael Di Stefano [16] defined grid computing as follows:

"Grid computing is any distributed cluster of compute resources that provides an environment for the sharing and managing of the resource for the distribution of tasks based on congurable service-level policies."

While Ian Foster brought three features of a grid in [4]:

- *Coordinates resources that are not subject to centralized control:* A Grid integrates and coordinates resources and users that live within different control domainsfor example, the users desktop vs. central computing; different administrative units of the same company; or different companies;

- *Using standard, open, general-purpose protocols and interfaces:* A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access.

- *To deliver nontrivial qualities of service:* A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security... to meet complex user demands.

Today, many people still confuse the concept of "grid computing" and "P2P computing". Both of them have common features in solving the problem of sharing resources between virtual organizations [6], and are based on layered architecture. Each approach has strengths and weaknesses in particular. Grid is built on strong architecture, serving small-scale user community, usually in the same field of study that they interest in. Grid support the ability to integrate diverse resources more powerful than P2P computing. Applications on grid is often scientific ones, requiring high computing power and user oriented applicatioins. Conversely, P2P computing [5] is built on a more flexible architecture to serve a large number of users, could reach millions. It primarily support resource sharing applications such as sharing computing cycles or file.

2.2 Data grid

2.2.1 What is data grid?

Physically, grid includes a large number of computers or clusters linking together via a network, being managed by efficient management techniques. In terms of logic, grid computing comprises two components: the compute grid and the data grid (Figure 2.1).

Grid computing: provides work managing services on distributed environment, resources are spare CPU cycles, network bandwidth, memmory...

Data grid: provides data management, access, synchronization, and data distribution in grid. The resouces of data grid include data storage systems, network

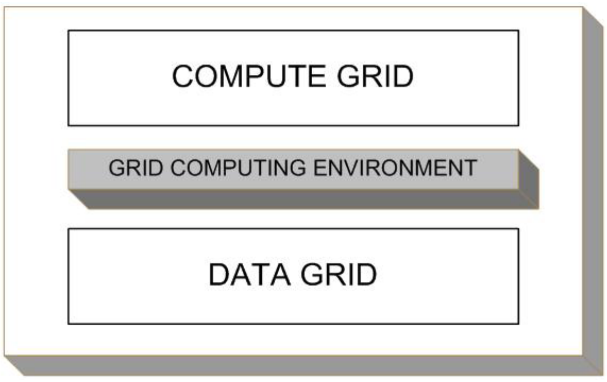

Figure 2.1: The components of a grid

bandwidth and files distributed on the grid nodes. Grid was developed to solve the problem of parallel which requires analysis on large data sets. The data are usually static and stored as files. These data are distributed on different nodes of grid computing. How do the computing nodes access this resources effectively? How to ensure the integrity and data synchronization? ...Data grid will answer above questions.

In any grid environment, data management are essential because the most important resource of the application is data. A question arises: why we do not use current data management architecture like database to solve data management in grid environment? The answer is: because the distribution, not heterogeneous are features of grid, so traditional solutions are not applicable.

2.2.2 Classifying Data grid

Data grid are used to solve the problem with data distribution over the grid environment. The data can be static, not change frequently and often in scientificapplications such as weather modeling and prediction [12], quantum analysis, gene map...

However, we also see applications whose input data change frequently each day or each hour. This class of applications includes earthquake warning, weather prediction, storms forecast systems... In these systems, the sensors are located in many places, they continuously measure data and send back to data grid for processing and data analysis, from which making appropriate decision and alterfication. We also can see dynamic data set in stock-market analyzing applications in which the rate of change index are very high throughout the world. For these type of applications, ensuring uniformity, handling update is crucial with the system. It is well known that data grid can be devided into two types depending on static or dynamic nature of the data sets.

Data grid level 0

Data grid level 0 provides the ability to manage large static data sets distributed over the grid nodes. This is data grid type which was researched and deployed early in the world. Because the data does not change over time, data grid level 0 can not solve problems relating to updateing, transaction management, data integration with external systems. Data grid level 0 was mentioned by Chervenak and colleagues in [2].

"In an increasing number of scientific disciplines, large data collections are emerging as important community resources. In this paper, we introduce design principles for a data management architecture called Data Grid. We describe two basic services that we believe are fundamental to the design of a data grid, namely, storage systems adn metadata managementl. Next, we explain how these services can be used to develop higher-level services for replica management and replica selection. We conclude by describing our initial implementation for data grid functionality"

On data grid level 0, the most important issue is moving data sets between grid nodes. Currently, there are three basic techniques are used:

File Transfer Protocol (FTP): are used popularly for data transmission on the network. In the grid environment, the functions provided by FTP are not enough; therefore, Globus project proposed and develop a new protocol GridFTP [18], which support data transfer better. GridFTP is suitable with large data sets which do not change much overtime.

Distributed File System (DFS): also managing distributed file over the network. From the user's perspective, DFS likes a directory tree format. Users do not need to know which node does the file actually locate. The architecture of DFS is based on the server. There is a server managing mapping between the physical file names on a particular node and logical file names in a directory tree. Users browse folder tree and send requests to the server. The server negotiate with the node which stores corresponding physical file, then send the actual file to the user. An example about DFS is CODA [3] or Network File System. However, due to strong dispersion, heterogeneous and not subject to centralized control, DFS is rarely used.

Metadata Center: integrating data from different sources and formats into a consistent structure which is suitable to the required format of applications. Metadata Center is the intermediate layer locating between applications and data sources. This layer's task is covering the complex techniques associated with connection, format transformation, and data transmission. This is resemble to the concept of Virtual Data Grid was mentioned by Ian Foster in [8].

The common between three above techniques is not supporting transaction management and update handling. They only fit for invariant data sources and little change over time.

Data grid level 1

Data grid level 1 supports dynamic data set which frequently changing over time. Besides the normal functions, data grid level 1 must be able to provide transaction management and data synchronization capabilities. Three popular techiniques are used to build a data grid level 1: JavaSpace, Distributed Memory, and Global Replication.

JavaSpace [9] is a distributed programming architecture in which distributed processes communicate with each other by passing objects through "space" or shared memory, rather than direct communication via message passing or remote method invocation. A process can "write" to and "read" from a JavaSpace, but it cannot modify an object while it is in a JavaSpace. For an object to be modified, it must first be removed from the space, changed, modify, and then put back into the space. Figure 2.2 give us a better perspective on this issue. Aditional properties of JavaSpace include:

- *Persistence.* An object read into a JavaSpace is persiatent in the space until it

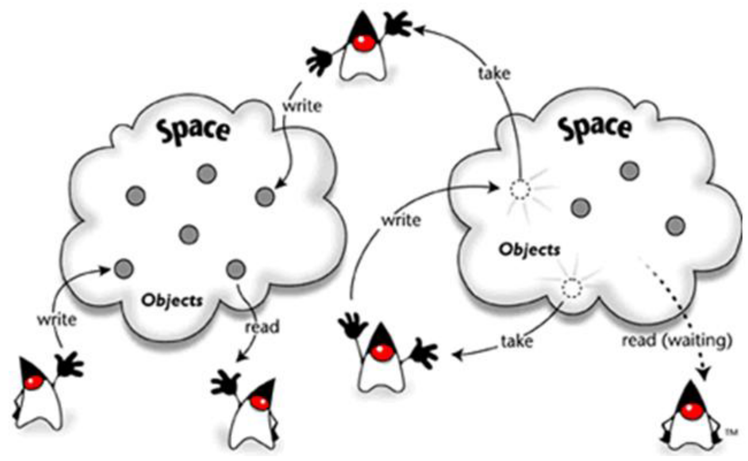

Figure 2.2: Collaboration between distributed processes in JavaSpace

is removed.

- *Transactions.* Operations into and out of the space, as well as those where spaces interact with each other are transactional.
- *Query.* JavaSpace supports a filtering or associative lookup mechanism where queries can be made to the space on the basis of a filter or template; the space will return objects that meet that filtered criteria.

These features are appropriate to build a data grid level 1. Because JavaSpace shares objects, the attributes and executable methods of the objects are also shared. Hence, JavaSpace exhibits characteristics of a compute grid. However, distributed applications using JavaSpace must be written in the Java language. Distributed applications that do not support Java cannot participate without gyrations of interfaces and bridges into the space.

Global Replication is a process of providing distributed and durable storage of data across a large grid of computers. On such system is the OceanStore project. Data objects are distributed and stored locally on the nodes of the grid. OceanStore's storage and replication algorithms are such that they ensure strong consistency among the object replicas. However, realizing that not all applications will have the same requirement for strong consistency and fault tolerance, global replication also allows applications to move to a weaker consistency policy that will boost performance. One more interesting aspect of OceanStore is its ablility to manage *data affinity*, the ability to move data to the nodes that most frequently use these data, keeping them cached locally to that node and thus reducing network traffic and increasing overall performance. However, like JavaSpaces, the global replication method is currently being prototyped only in Java, thus introducing the issues of integrating non-Java applications.

Distributed Memory. Distributed and shared memory is an effective mechanism for building a data grid. OpenMP [11], although not designed for grid computing, is designed for the splitting of large processing loops into smaller bits of work in a multithreaded, multiprocessor environment like SMP - Symetric Multiprocessor. It also creates a distributed memory space to eliminate the use of traditional network communication methods and middleware to allow the threads to share data. Sound a lot like grid computing, or does it?. Some features of OpenMP are: compiler-based, thread-based, shared memory, distributed memory.

2.2.3 Features of the data grid

Data grid is often compared to other methods of traditional data management, especially relational data management, so that they can see the hightlight features of data grid than previous techniques.

The traditional data management techniques

Of all data management techniques, relational data management is the most common technique used. The characteristics of this technique are shown in a number of ways:

Data engine: the intermediate layer connecting between logic aspect with the physical aspect of data. The model consists of a data governance program and a storage device management program (Figure 2.3). The characteristics of data engine decide storage speed, access speed, replication speed, scalability as well as efficiency of the system.

Data structure: data are organized into two-dimensional tables, each row is a

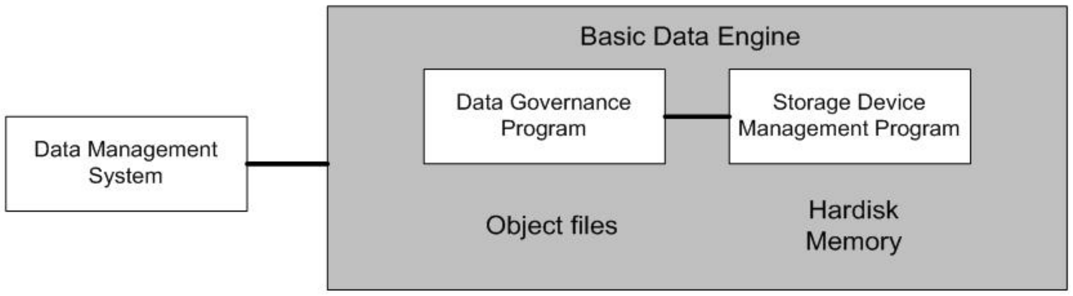

Figure 2.3: Data management system and its data engine

record, and each column is an attribute of the record. Table are mapped to files on disk through the mengine (Data engine). The database management system allows complex relationships between tables. Uniqueness of records in the table is set by key. The tables which have common attributes can be linked together through a foreign key.

Access: Access to the database management system is done by a structured query language called SQL. SQL query language is a powerful, simple, easy to learn and use. With SQL, users can:

- Create data schemes
- Query tables' data
- Create relationships between tables.
- Modify table by *update* and *delete* command

Through the SQL language, users can easily access the system. This is one of many reasons explains why relational data management are widely used.

Integrity: when updating or adding data, application have to ensure correctness and accuracy of the data. The relational database management system are solely responsible for ensuring data integrity. There are two levels of integrity: record integrity and referential integrity.

Transaction management: comprises a unit of work performed within a database management system, and treated in a coherent and reliable way independent of other transactions. Transactions in a database environment have two main purposes: *to provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure* and *to provide isolation between programs accessing a database concurrently.*

Event management: the database management system supports event management, both inside and outside the system. When events occur, the system acts in accordance with the event. For example, when users delete data on a table with reference to some other tables, the corresponding data in these tables should be deleted.

Backup, recovery and availability: the database management system must be a reliable information resources. It stores knowledge for operation of professsional units. Therefore, it must be able to recover quickly when errors occur. To ensure this, backup is one of effective measures. Besides, the availability of data to meet the requirements of business applications is also important criteria. Some techniques to do this is replication, tracking, backup parts or all.

Security: in a relational database management system, security is crucial. Database management system has a machanism to set users' authority. Data is only visible to authorizied users. User authorization, group, encryption are most popular techniques to ensure the database security.

I presented some technical features about traditional database management system (DBMS). We can see that there are two main reasons make DBMS widely used: *easy to use* and *consistency.* Detail discussion about DBMS is presented in [10].

From relational database to data grid

Relational database management system is a good fundamental knowledge when studying about the development of data management in grid environment. Users, professional applications can easily access relational DBMS through friendly interface - SQL language. Techniques like security, transaction management, integrity make data always in consistency and ready.

Although almost DBMS are built basing on relational model, it is not suitable model in grid which is distributed and heterogeneous environment. Data on grid are often not structure, large size and dispersed thousands of computer distributed geographically. Therefore, we need a data management architecture which is suitable with distributed environment's features called data grid.

Figure 2.4 links a baseline of data grid vocabulary to well-known relational database terms. Relational database implementations have two fundamental components: (the underlying engine that manages physical resources, in this case a disk and (2) a layer on top of that to provide all the data management features and functionality that architects and developers whould rely on for data management, querying, arrangement of data in highly ordered structures such as tables, the ability to transact on data, leveraging stored procedures, event triggerings, and transating in and out of the database with external systems. These are the management features and functions that today are where our true interest lies. How do I manage tables/row locking? How do I structure indices for maximum performance? Very little attention today is given to the underlying engine.

In the same way that *relational database* is a generic term, so is *data grid*. Conmpanis will offer implementations, products of their vision of what a data grid is. To analyze the differences between the products offered, it is possible to apply a baseline consisting of generic term, implementation, data management, and engine. Each implementation of data grid will have an engine. That engine may be a metadata dictionary or a distributed cache. It will also handle the data management aspects of this data grid, defining how to structure data in tables, arrays, or matrices; how to query data; and how to transact on the data.

Depending on the exact implementation of this enginewhether it is a metadata dictionary that routes requests to the true long-term persistent stores, or a distributed cache that spans all computers in the grid to form one virtual spacethere are specic data management issues for this new topology. How to synchronize, how to transact on the data, how to address data afnity? These are all data management issues; issues that, no matter who the architect or application developer is, will need to be addressed within their applications. These are the quality-of-service (QoS) levels that are required of the data grid. If a data grid does not provide such service, then developers will have to write down to the lowest, most fundamen- tal level of bit and byte management.

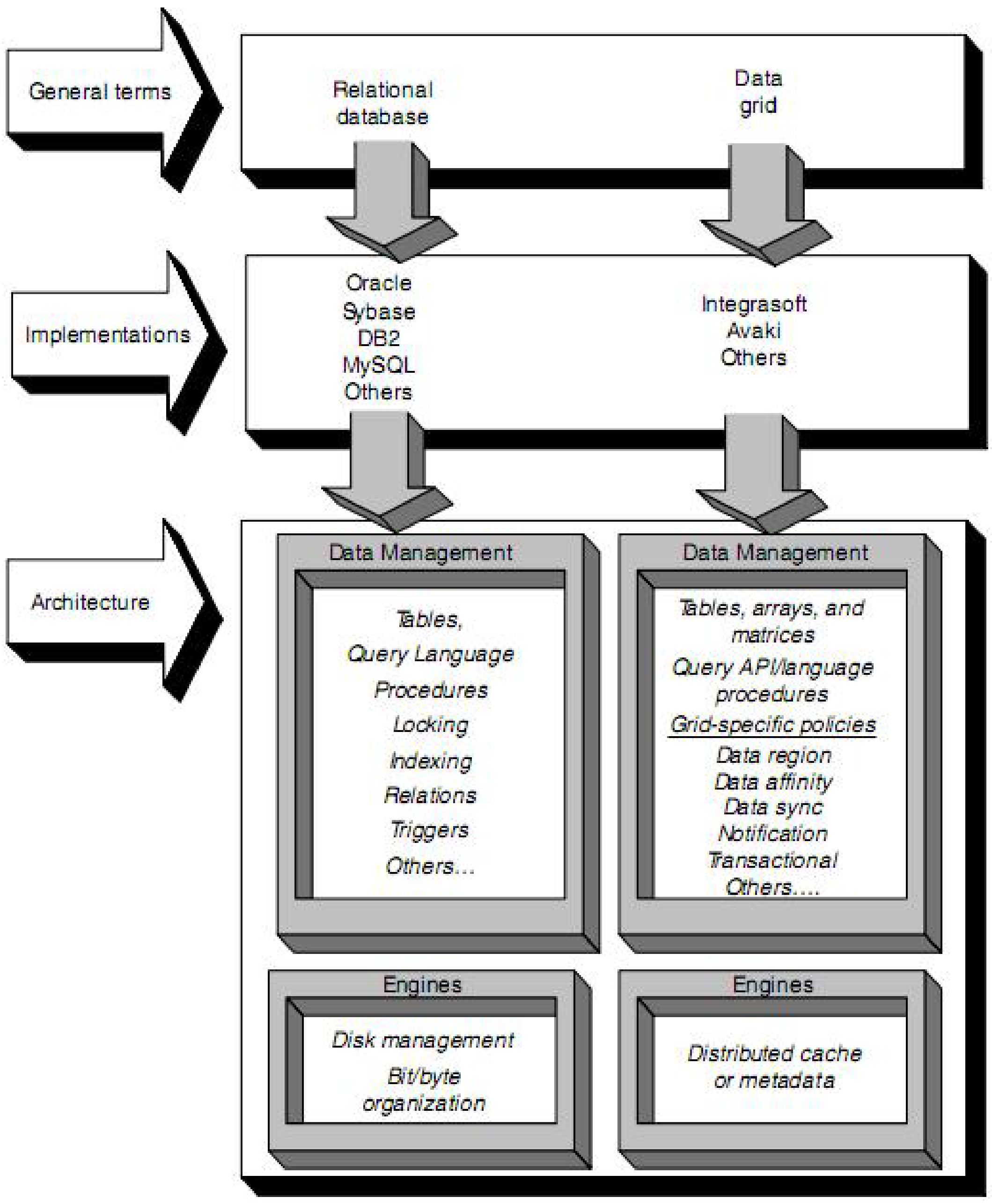

Figure 2.4: Baseline of terms and function

Data grid support for true data management extends to facilitation of the adoption and widescale acceptance of grid technology. Developers can easily transit from client/server-based applications to a grid topology by leveraging a product that provides the same levels of service quality that have become the standard with relational databases.

Requirements of a data grid

With new technology come new features that are required and the associated data management specific to grid technology. Services specific to data management in grid environments include data regionalization, data synchronization policy, transactional data policy, coordination of task scheduling to data locality, which are discussed in the sections that follow.

Data Regionalization. Regionalization of data within a grid is a key performance and management feature that is seldom available within other infrastructures. A data region within a grid is an organization of data that spans machines and potentially geographies and caters to the needs of the users of that region. Region is the highest level of constructing the data in the data grid. Region drives the aggregation of the data and the policy instructions regarding the data. For example, a data region is analogous to the database in the relational model. It contains other structures of data or data schema specic to a line of business.

Data Synchronization. Data synchronization typically falls into two spectra: strong synchronization and weak synchronization. Strong synchronization policies enforce a tight replication of like patterns of data among the nodes of the data region as well as strictly enforcing the replication policies among the nodes of the data region. The strong synchronization mechanism is used when low latency and high consistency are required from the data grid at the cost of scalability and exibility. Weak synchronization policies, on the other hand, enable data to be synchronized on an as needed basis and sometimes not at all. The weak synchronization mechanism allows for less data consistency but for higher scalability and exibility.

Transactional Data Synchronization. Transactional data synchronization is very important even in the data grid since the ability to recover, commit, roll back, and the like are important to data integrity. Transactional data policies fall into basic categories within a data grid: *optimistic* and *pessimistic*. *Optimistic* transactions are typically implemented with little or no locking and coordination; they are transactions only in the sense that in a possible conict situation, an exception is thrown and the user is notied. *Pessimistic* transactions are typically multiphase with the proper locking, unlocking, commit, and rollback that are typical for transactional integrity. The mechanism used is that of XA, which is associated with the destination, and all transactional management (e.g., all locking) is done through that destination.

Data Locality (Data Distribution). The ability to locate and group data depending on data usage is essential to the data grid from a performance perspective. Data locality thus can be dened as the clustering of data depending on usage. This feature, which is specic to the data grid, enables the architectures to scale signicantly better than did previous technologies architectures. The data grid implements data locality through two sets of synergistic architectures:

- Data within a data grid are associated with a locality that pinpoints the exact resource that owns a primary copy of the data and its neighboring topology (this information is provided through the APIs to the architecture using the data grid in order to *propagate and distribute work to the particular resource*)

- Metrics of data usage improve the availability of the data in the data grid. Data are monitored through a set of metrics, and *individual data blocks can migrate from resource to resource* depending on the usage patterns and history of use.

The migrating data and distributing work depending on cost criterias on the data grid.

In the paper *"Distributed Computing Economics"* [7], Jim Gray presents some criterias which affect data or work migration including network resource cost, computing cost and storage sysem access. For techniques about data migration and work distribution, Ian Foster contribute an important research [1], in which he and his collegues build a scheduler comming along with data migration algorithms based on some main measures including average respond time of a job, average CPU idle time...

Digging down into these problems is intereting research path but beyond the thesis. Detail information about above requirements can be found in [17].

williamstallingscomputerorgandarchitecture

2.2.4 Architecture of data grid

Data Grid Design

Data grid applications must frequently operate in wide are, multi-institutional, heterogeneous environments. According to Ian Foster [16] (page 3), data grid design have to follow four principles:

Mechanism neutrality. The data grid architecture is designed to be as independent as possible of the low-level mechanisms used to store data, store metadata, transfer data, and so forth. This goal is achieved by defining data access, third-party data mover, catalog access, and other interfaces that encaptulate peculiarities of specific storage systems, catalogs, data transfer algorithms, and the like.

Policy neutrality. The data grid architecture is structured so that, as far as possible, design decisions with significant performance implications are exposed to the user, rather than encapsulated in "black box" implementations. Thus, while data movement and replica cataloging are provided as basic operations, replications policies are implemented via higher-level procedures, for which defaults are provided but that can easily be substituted with application-specific code.

Compatibility with Grid infrastructure Data grid arhitecture integrates with grid infrastructure to overcome the difficulties of wide are, multi-institutional operation by exploiting underlying Grid infrastructure that provides basic services such as authentication, resouce management, and information.

Uniformity of information infrastructure As in the underlying Grid, uniform and convenient access to information about resouce structure and state is emphasized as a means of enabling runtime adaption to system conditions. In practice, this means that using the same data model and interface to access the data grid's metadata and replica catalogs.

From above design principles, data grid must have layered architecture. The lowest layer consists of basic services. The upper services on higher layer are basing on these basic services.

Besides, storage and synchronization policies must be mentioned. Whether data grid is designed according to *replication-oriented* or *distribution-oriented.* In *replication-oriented* architecture, each member of the data grid has a cache storing all data in the grid. Whenever cache data of a member is changed, updated information is propagated to other caches. This is strong synchronization which was mentioned above. This architecture is suitable to grid which requires high reliability and integrity, but low performance because of the cost for synchronization between grid nodes. Hence, this type of data grid is not scalability.

Inversely, *distribution-oriented* architecture shares data basing on peer-to-peer network topology. In this architecture, data is not replicated at all the nodes. However, a certain level of replication is also ensured. For example, a file can be replicated to 10 copie stored on 100 grid nodes. How many copies are replicated on each nodes depending on performance, usage policy and particularity of data grid. Scalability capability of this architecture is much more higher than replication-oriented architecture. We can find the similarity of this architecture with RAID model - Redundant Array of Inexpensive Disk [15], data is dispersed on an array of disks. The disadvantage of *distribution-oriented architecture* is low reliablity and data integrity.

Discussing about data integrity, there are two strategies to ensure this. Firstly, data grid has a centralized management component which is responsible for synchronizing data on all grid nodes. Such program is hard to build carrying much weeknesses such as less scalability or single-point failure.

Chapter 3

Construction of data grid for GOODAS system

Chapter 4

Implementation and evaluation

Chapter 5

Conclusion and future development

Appendix A

A Long Proof

A.1 Qoh.1

Section 1

A.2 Qoh.2

Section 1

Bibliography

[1] Y. Gil C. Kesselman G. Mehta K. Vahi K. Blackburn A. Lazzarini A. Arbree R. Cavanaugh S. Koranda [17]. E. Deelman, J. Blythe. *Mapping Abstract Complex Workflows onto Grid Environments.* Journal of Grid Computing, 1(1), 25-39, 2003.

[2] Carl Kesselman Ann Chervenak, Ian Foster. *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets.* 2003. page 1.

[3] Peter J. Braam. The coda distributed file system, http://www.coda.cs.cmu.edu/.

[4] Ian Foster. *What is the Grid? A Three Point Checklist.* GRIDToday, 2002. Page 2-3.

[5] Ian Foster and Adriana Iamnitchi. *On Death, Taxes, and Convergence of Peer-to-Peer and Grid Computing.* In 2nd International Workshop on Peer-to-Peer Systems. Berkeley, CA, 2003.

[6] Iran Foster and Carl Kesselman. The anatomy of the grid, enabling scalable virtual organizations, 2002.

[7] Jim Gray. *Distributed Computing Economics.* Microsoft Research, CA, 2003.

[8] Michael Wilde Ian Foster, Jens Vockler and Yong Zhao. *The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration.* 2003. Page 1.

[9] Part I Ease the Development of Distributed Apps with JavaSpaces Make Room for JavaSpace.

[10] Kim Anh NGUYEN. *Principles of database management systems.* Hanoi National University. 'Chapter 3, 5, 7, 8.

[11] OpenMP. http://www.openmp.org/drupal/.

[12] Earth System Grid Project. http://www.earthsystemgrid.org.

[13] Globus project. http://www.globus.org/toolkit.

[14] Borja Sotomayor. *Globus toolkit 4 programming java services.* Morgan Kaufmann, Inc, 2006.

[15] William Stallings. *Computer Organization and Architecture.* 5th, Prentice Hall Inc, 1996. Chapter 5.

[16] Michael Di Stefano. *Distributed Data Management for Grid Computing.* John Wiley & Sons, Inc, 2005. Chapter 1, page 4.

[17] Michael Di Stefano. *Distributed Data Management for Grid Computing.* John Wiley & Sons, Inc, 2005. Chapter 9-12.

[18] R. Kettimuthu M. Link W. Allcock, J. Bresnahan. *The Globus Striped GridFTP Framework and Server.* Proceedings of Super Computing 2005 (SC05), Nov 2005.