Knowledge based Replica Management in Data Grid Computation

Riaz ul Amin[1], A. H. S. Bukhari[2]

[1]Department of Computer Science
University of Glasgow
Scotland, UK
[2]Faculty of Computer and Emerging Sciences
Balochistan University of Information Technology, Engineering and Management Sciences
Quetta, Pakistan.

riaz@dcs.gla.ac.uk, ahsbukhari@yahoo.com

Abstract. As the number of scientific disciplines has increased, the large data collections are emerging as important community resources. In domains as high energy physics, and computational genomic, the volume of interesting data is already measured in terabytes and will soon total peta bytes. The Research communities need to access and analyze this data using complex computational and access techniques. No data management infrastructure provides solution against the complex computational analysis of huge and geographically distributed data sets. Most of queries usually search analyzed data from terabytes of distributed data repository, over wide area networks. Replicas, and other advanced techniques collectively maximize the use of scarce storage, networking, and computing resources. Existing data grid replication technique no doubt provide us with availability of required data sets, but in order to create replica it has to bear an overhead of huge computation for required data sets. Large network traffic cause the performance unsatisfactory. Our goal in this effort is to provide users with replication infrastructure in Grid that uses Knowledgebase having learning capability so as to reduce the computation for creating dataset on each user request.

Keywords: Data Grid, Knowledgebase, Replica Management, Replica Consistency, Ontology, KRE, Relational DBMS, Meta data Services.

1. INTRODUCTION

Replica management is the grid service and data grid has replica manager to create or delete the replicas in storage systems. The basic purpose of creation of replica at some location is to respond the user request quickly but achieving this benefit we have to take care about the management of storage and network traffic. Here we have proposed data grid framework for replica management introducing the Knowledge representation Engine with its outcome as knowledgebase, which can be replicated in less time and storage space as compared to the original datasets.

2. DESIGNED SYSTEM ARCHITECTURE

Following are main components of the designed architecture used for knowledge base replica management in data grid computing. Each component is described in detail:

2.1. General Data Grid

Generally data grid consists of two levels of services, low level services and high level services. We have tried to improve one component of low level services i.e. Replica Management using knowledgebase representation. We are not aware of how the knowledge is represented in human mind till with use of knowledge representation techniques and methods, we are able to process our data forming the knowledgebase more over the implementation of ontological rules helps us to formulate an exhaustive and rigorous conceptual schema within a given domain, a typically hierarchical data structure containing all the relevant entities and their relationships and rules (theorems, regulations) within that domain.

2.2. Replica Structure

Replica is exact copy of a file having well defined link with original file. Replica may be consistent read only if want a change we have to generate a new file. It may be consistent read/write replica, which are modifiable and replica may be versioning that is the combination of read only and read/write replicas.

Replica Manager. Its purpose is to manage all replica management related functions. For the implementation of replica manager we purposes SOAP (Simple Object Access Protocol) based system that replicate the required information gained from knowledgebase. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelop that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing Remote Procedure Calls and responses.

Replica Location Service. It records the information related to physical storage locations of the logical identifiers.

The Updating Methodology. In our proposed system there is no problem with synchronization of replica and the master file as the knowledgebase Engine inquires the master file status after the limited time and so can update the replica as the time passes.

Replica Consistency. As discussed earlier replica's consistency is assured by KRE' Inquiries from master file.

2.3. KRE's Functions

The knowledge representation engine (KRE) was build to generate information space tree as follows:

- Meaningful terms in request of grid client using entropy
- Index is calculated
- Heuristic and ontological rules are applied

KRE has capability of learning. It inquires the master file after the specified time and make its knowledge up to date and so we are able to update our replica updated and consisted. The KRE also keeps the information about metadata.
Previously the replica management services have to register three types of entries in a replica catalog or repository:

Logical Files. These files are with unique global name but may be more than physical existences. There is unique entry for each physical existence in knowledgebase.

Logical Collections. These are user defined groups of files. Logical collection is a user-defined group of files without knowing where they are stored.

Locations. Location entries in the replica management system contain all information required to map a logical collection to a particular physical instance of that collection. This might include such information as the hostname, port number and access protocol of the physical storage system where the files are stored. Each location object represents a complete or partial copy of a logical collection on a storage system. One location entry corresponds to exactly one physical storage system location.

When replica is created all the required files are accessed and replicated as a whole even that data that don't need to be replicated, as there was no mechanism that is helpful to decide about the data to be replicated. We have introduced the layer Knowledgebase and Knowledge Representation Engine. KRE after performing required computation represents the resulted data to be replicated in the form of knowledge and Replica Manager Module takes the replica of this knowledge and knowledgebase is updated.

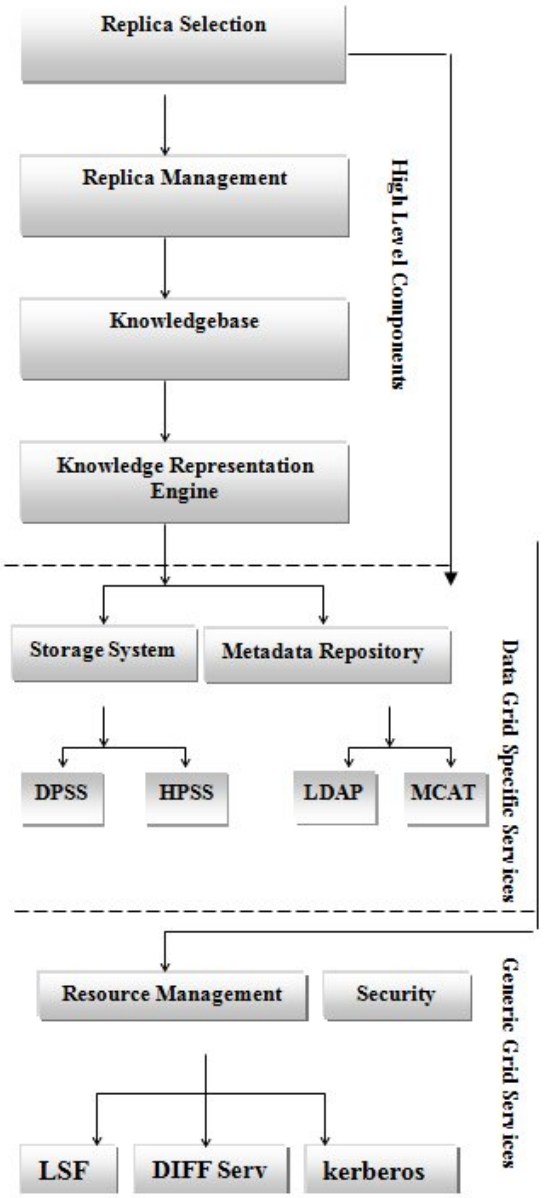

Figure 1. Workflow of the designed system

2.4. Support for new Operations

The proposed architecture for knowledge based replica management provides the support for following operations:-

- Modify / Update Knowledgebase
- New Entry In Knowledgebase
- Replicate Knowledgebase
- Deletion of Replica

3. IMPLEMENTATION DETAILS

For the replica management service architecture we have described, there are many possible implementations. In this section, we discuss a few implementation issues.

3.1. Storing and Querying Replica Information

There are a variety of different technologies that can store and query replica management service information. Two possibilities are relational databases and LDAP directories. Where LDAP, Lightweight Directory Access Protocol, is an Internet protocol that email and other programs use to look up information from a server.

A relational database provides support for indexing replica information efficiently and for database language queries (e.g., SQL) of replica management information. An LDAP or Lightweight Directory Access Protocol directory has a simple protocol for entering and querying information in a directory; an LDAP directory can use a variety of storage systems or "back-ends" to hold data, ranging from a relational database to a file system.

3.2. Reliability and Availability

This is an implementation decision that if the replica management service fails during an operation, it will be unavailable for some time and its state upon recovery will be indeterminate. As with any system design, there is an inevitable tradeoff between the level of reliability and consistency after failures and the system's cost and complexity.

Reliability and availability will be greatly improved in implementations that replicate and/or distribute the replica management service. Our architecture allows implementers to use services provided by relational databases and LDAP directories for distributing and replicating information. If high reliability and availability are required, then system builders must devote adequate resources to replicating the service, performing frequent checkpoint operations to facilitate quick recovery, and

avoiding single hardware and software points of failure. This robustness must be engineered into the replica management service.

4. CONCLUSION

We have argued that data-intensive, high-performance computing applications require the efficient management and transfer of terabytes or peta bytes of information in wide-area, distributed environments. Researchers performing these analyses create local copies or replicas of large subsets of these datasets to overcome long wide-area data transfer latencies.

A Data Grid infrastructure, to support these applications must provide a set of orthogonal, application-independent services that can then be combined and specialized in different ways to meet the needs of specific applications. These services include a meta-data management, replica management, replica selection service, and secure, reliable, efficient data transfer. We have presented architecture for a replica management service. The new feature presented in this paper is that it contains two extra layers i.e. KRE and Knowledgebase layer. The replica management service can be used by higher-level services such as replica selection and automatic creation of new replicas to satisfy application performance requirements.

5. FUTURE WORK

This is the basic level architecture that we have presented but a lot of work has to be done in order to design the architecture of Knowledgebase and learning process of this knowledgebase.

REFERENCES

1. Ann Chervenak,Ian Foster(The Data Grid towards architecture for distributed management and analyses of large scientific datasets)
2. M. Baldonado, C. Chang, L. Gravano, and A. Paepcke. The Stanford digital library metadata architecture. Intl J. Digital Libraries, 1(2):108{121, 1997.
3. Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. The SDSC storage resource broker. In Proceedings of CASCON'98 Conference. 1998.
4. M. Beck and T. Moore. The Internet2 distributed storage infrastructure project: An architecture for internet content channels. Computer Networking and ISDN Systems, 30(22-23):2141{2148,1998.

5. Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. The extensible markup language (xml) 1.0. W3C recomendation, World Wide Web Consortium, February 1998. ee http://www.w3.org/TR/1998/REC-xml-19980210.

6. S. Cousins, H. Garcia-Molina, S. Hassan, S. Ketchpel, M. Roscheisen, and T. Winograd. Towards interoperability in digital libraries. IEEE Computer, 29(5), 1996.

7. Renato Ferreira, Tahsin Kurc, Michael Beynon, Chialin Chang, Alan Sussman, and Joel Saltz.

8. OGSA http://www.globus.org/ogsa/

9. OGSI http://www.gridforum.org/ogsi-wg/

10. S. Burbeck, "The Tao of e-Business Services," IBM Corporation (2000); see http://www-4.ibm.com/software/developer/library/wstao/index.html.

11. G. Agrawal. High-level Interfaces for Data Mining: From O_ine Algorithms on Clusters to Streams on Grids. In Workshop on Data Mining and Exploration Middleware for Distributed and Grid Computing, Minneapolis, MN, September 2003.

12. Imran Sarwar Bajwa, "Middleware Design for Computing Cluster to Process Satellite Data", International Conference on Computers & Emerging Technologies (ICCET 2011), Pakistan, pp:112-116, 2011

13. Kamal Ali and Wijnand Van Stam. TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture. In Proceedings of The Tenth ACM SIGKDD Conference (KDD'04), Seattle, WA, August 2004.

14. N. Amado, J. Gama, and F. Silva. Exploiting Parallelism in Decision Tree Induction. In Parallel and Distributed computing for Machine Learning. In conjunction with the 14th European Conference on Machine Learning (ECML'03) and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03), Cavtat-Dubrovnik, Croatia, September 2003.

15. V. S. Ananthanarayana, D. K. Subramanian, and M. N. Murty. Scalable, Distributed and Dynamic Mining of Association Rules. In Proceedings of HIPC'00, pages 559–566, Bangalore, India, 2000.

16. H. Andrade, T. Kurc, J. Saltz, and A. Sussman. Decision Tree Construction for Data Mining on Clusters of Shared Memory Multiprocessors. In HPDM: High Performance, Pervasive, and Data Stream Mining 6th International Workshop on High Performance Data Mining: Pervasive and Data Stream Mining (HPDM:PDS'03). In conjunction with Third International SIAM Conference on Data Mining, San Francisco, CA, May 2003.

17. E. Ariwa, M. Senousy, and M. M. Gaber. Facilities Management and E-business Model Application for Distributed Data Mining using Mobile Agents. The International Jour- nal of Applied Marketing, 2(1), 2003.

18. Ezendu Ariwa and Medhat Medhat Gaber. Globalization and Informatization: Analysis of the Application of Distributed Data Mining to Facilities Management. In 32nd International Conference on Computers and Industrial Engineering Sustainabil- ity, Globalisation-The Engineering Challenge, 2003

19. J. Aronis, V. Kolluri, F. Provost, and B. Buchanan. TheWoRLD: Knowledge Discovery from Multiple Distributed Databases. Technical Report ISL-96-6, Intelligent Systems

Laboratory, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 1996.

20. M. Z. Ashrafi, D. Taniar, and K. A. Smith. A Data Mining Architecture for Distributed Environments. IICS 2002, pages 27–38, 2002.

21. R. Agrawal and J. C. Shafer. Parallel Mining of Association Rules. IEEE Transactions On Knowledge And Data Engineering, 8:962–969, 1996.