A Scalable Multi-Replication Framework for Data Grid

Shin'ichiro Takizawa
Tokyo Institute of Technology
takizawa@matsulab.is.titech.ac.jp

Yasuhito Takamiya
Tokyo Institute of Technology
takamiya@matsulab.is.titech.ac.jp

Hidemoto Nakada
National Institute of Advanced Industrial Science and Technology
hide-nakada@aist.go.jp

Satoshi Matsuoka
Tokyo Institute of Technology & NII
matsu@is.titech.ac.jp

Abstract

Existing replica services on the Grid we know to date assumes point-to-point communication and file transfer protocol. As such, when hundreds to thousands of hosts on the Grid access a single dataset simultaneously, bottlenecks in networks and/or the data servers will hinder performance significantly. Instead, our replication framework couples efficient, multicast techniques with a replica catalog that automatically detects simultaneous access to the replica by multiple nodes. As a prototype, we have designed and built a portable, XML-based replica location service accounting for such parallel transfer requests, and coupled it with a O(1) bulk file transfer system Dolly+[6]. The benchmarks show that the system is scalable and effective in reducing replication costs significantly in cluster-based replication scenarios.

1. Introduction

Various scientific disciplines, such as high-energy physics, astronomy, and lifesciences, are now building and deploying Grids in order to process the datasets collected through their observational instruments, typically referred to as Data Grids. The sizes of such datasets on a Data Grid could be quite large, ranging from a few terabytes up to several petabytes, and moreover, they are typically accessed remotely over high-speed networks as researchers and their computing facilities on the Grid are globally distributed. In order to facilitate their efficient and safe management, various work have focused on algorithms and systems for managing replicas on Data Grids: for example, The Globus toolkit facilitates a *replica management service*[1] that manages the location of the datasets and their replicas in a hierarchical fashion in a database called the *replica catalog*[2], and transfers the replicas using GridFTP[3], one of the standard file transfer services on the Grid. The Grid DataFarm[4] is a parallel and distributed filesystem on the Grid, and utilizes replicas extensively to achieve efficiency and data fault tolerance.

Existing replica services on the Grid we know to date assumes point-to-point communication and file transfer protocol. As such, when hundreds to thousands of hosts on the Grid access a single dataset simultaneously (a typical scenario for many applications), bottlenecks in networks and/or the data servers will hinder performance significantly. As an example, in the BLAST[5] DNA sequencing application, the DNA sequence database must properly replicate amongst all compute nodes, and such replication may happen frequently due to rapidly changing datasets of newly acquired DNA sequence data, rendering common optimization techniques such as caching useless.

Instead, we propose a replication framework for (Data) Grid that couples efficient, multicast techniques with a replica catalog that automatically detects simultaneous access to the replica by multiple nodes. More specifically, we have designed and built a portable, XML-based replica location service accounting for such parallel transfer requests, and coupled it with a O(1) bulk file transfer system Dolly+[6]. The benchmarks show that the system is scalable and effective in reducing replication costs significantly, especially in the case where the client peers of the replication are multiple nodes in a single cluster, accessing a remote data server.

2. Our Parallel Multi-Replication Framework on the Data Grid

Our parallel multi-replication framework on the Data Grid manages the location of data in the Grid, and employs efficient multicast techniques to drastically reduce the cost of replication when requests are made on a single data set in a parallel fashion from multiple nodes. Because the replica service has full knowledge of where the original data and the replica are, the requests that are made, and the network topologies and bandwidths of the relevant nodes in the Grid, the replica manager can select an appropriate strategy per each replica request, including the normal peer-to-peer high bandwidth transfer, to aggregating requests and employing fast O(1) multicast protocols when deemed advantageous. Our framework thus is most effective in a realistic setting, when a large-scale SPMD or parameter sweep applications involving hundreds of nodes issue frequent data access requests to an identical data set, as is with the case of the BLAST application.

Data replication for our framework is performed in two phases: the (1) data (originator) host selection phase, and the (2) file transfer phase, as depicted in Figure 1 and Figure 2.

1. The originator of the data registers its location and access info to the *data location management server*. The server hosts a data location service for the entire Grid for each piece of data, typically a file or a set of files.
2. The client inquires the location of the data it wishes to access to the file information management server.
3. The server in turn returns all the known locations of the particular data in question to the client.
4. The client selects the most appropriate data server (i.e., the originator of the data or a client hosting its replica) according to the topology and the status of the network, etc. (Note that at this point the decision on the data server is being made strictly locally.)

The file transfer happens in the following fashion as in Figure 2.

1. When multiple client hosts of some naturally definable group, typically in the same network site or within a same large-scale cluster, makes a data replication request within some short time interval, the request is made through a proxy or a representative node that dynamically aggregates the requests and sends the request to the data server.
2. The data server multicasts the requested data to all the representative hosts that requests the data using a efficient and reliable multicast algorithm to the representative nodes.

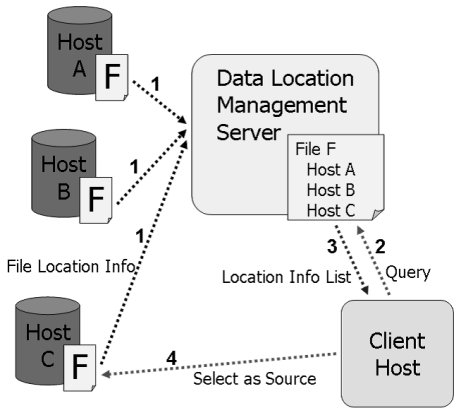

Figure 1. Selecting the Data Server Host in a Data Grid

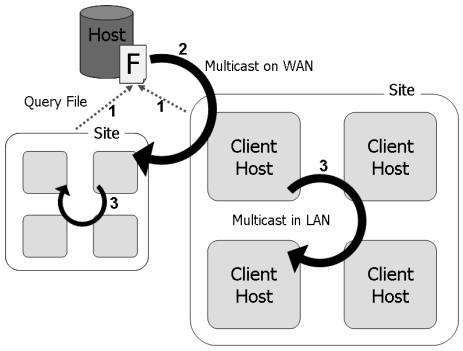

Figure 2. Data Replication using Parallel Multicasting

3. Each representative node in turn multicasts the data to all the hosts within the group.

Although the our framework itself is clean and simple as outlined, each component must satisfy the following requirements. Moreover, multicasting is typically considered as often not very reliable or scalable over hundreds of nodes in a network. In order to circumvent such problems, we facilitate an efficient and reliable O(1) multicasting we have devised for our Dolly+ system, but this must be shown to scale over numerous nodes in the network. Below, we investigate the requirements for the three components that serves as a key to our implementation of the framework:

Managing Data/File Location A given data and its replica may exist in numbers throughout a Data Grid, and the data location management server must manage and promptly return the choice in an efficient fashion. When multiple requests are made in parallel, this

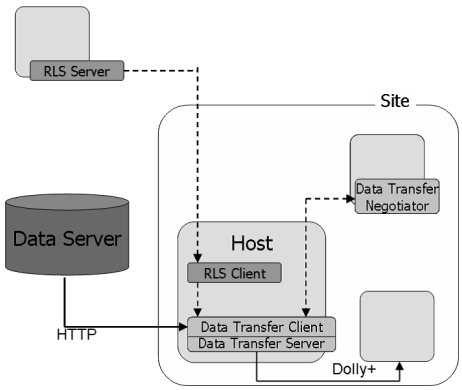

Figure 3. Overview of the Prototype System

server may itself may become a bottleneck. As such the server not only needs to be flexible but also requires high performance.

Network/Grid Monitoring The client in turn must select the most appropriate data server from the given list of candidates returned by the data location management server. Such a decision will have to be made based on the dynamic monitoring information of the Grid, in particular the status of its network. Although network monitoring for the Grid is considered as its essential constituent, we must nevertheless show that clients can make appropriate local decisions from its given info, and such must be shown to scale given the hundreds of nodes involved in the Grid and the aggregation of requests that happen therein.

Application-Level, Hierarchical Multicasting of Data As indicated above, our system uses application-level multicast for transfer amongst multiple 'groups' and the transfer within each group (i.e., typically a cluster with LAN and local storage). Because of significant differences in network characteristics for WAN and LAN, the system must shown to scale and exhibit O(1) scaling. Also important is to investigate the tradeoff point where our framework actually would be effective compared to conventional strategies such as caching within a local site.

3. Prototype Implementation of our Multi-Replication Framework

We have implemented a prototype of our framework to address the above issues. The system is written in Java, and the overall architecture is as shown in Figure 3.

3.1. Replica Location Service (RLS)

The RLS provides various services on data replica location including registration, update, deletion, search, etc. The server maintains data with its logical name (called the logical data or file name), and maintains the physical location and other attributes of the data in the replica catalog DB. Our RLS also supports logical collection services, a typical service often provided by other RLSs.

In our implementation we experimented with the use of a native XML DB Xindice[7] for flexibility. A logical collection is an entry in Xindice, and a logical data/file are the registered as its attribute, and the replica location is further defined as an attribute of the logical data/file. We added caching so that multiple parallel accesses from the nodes to the same entry will not cause significant conflicts. We have also employed XML-based protocol over HTTP for portability and firewall compliance.

3.2. File Transfer Service

Data Transfer Server A server that acts as a head node for multicast within a given group within a LAN. Although we have designed the prototype to accommodate various types of LAN file transfer services, we have employed the Dolly+ system we are developing for clusters that performs full-speed O(1) reliable multicast utilizing a ring topology. In order to aggregate as many transfer requests so that they can be multicast, we delay the startup of transfer for some adjustable time period.

Data Transfer Client The RLS client will select the best replica amongst all the available replicas (including the original data) from which the file will be transferred. The prototype is designed to hook into a Grid monitoring service to make the appropriate judgement. For our experiment, we simply took the RTT amongst the nodes as a metric, but in practice more comprehensive metrics should be used.

After making the selection the client behaves differently whether the host will be in the LAN or the WAN. If the host is within the same LAN, the request is sent directly to the data transfer service of that node. Otherwise, the request is sent to the transfer negotiator, which we describe below.

Data Transfer Negotiator The Transfer Negotiator aggregates multiple parallel requests from the same group of hosts within a LAN, thereby reducing external traffic outside the group. In our current implementation, the negotiator only grants request to fetch the replica data for the first node that requested the transfer; for other nodes, a reply is sent to fetch the replica locally, initiating multicast transfer amongst the nodes.

CPU	Mobile PentiumIII 1.13GHz
Memory	640MB
Network	100Base-T
OS	Windows XP Professional
Java	Sun JDK 1.4.2_01

Table 1. RLS Server PC

CPU	Athlon MP 1900+
Memory	768MB
Network	100Base-T
OS	Linux Kernel 2.4.19
Java	Sun JDK 1.4.2_03

Table 2. The Presto III Cluster at Titech

4. Performance Measurements of Multi-Replication

For brevity, we present the performance of our prototype system in a typical Data Grid scenario where a set of hosts within a cluster all request replica of a single remote dataset. The remote RLS server and the client PC cluster are described in Table 1 and Table 2, respectively. The replicated data is the entire image file of the Fedora Linux distribution (approximately 660MB), and as a remote server, we registered the Fedora's Australian mirror server (mirror.pacific.net.au) and a private server within our laboratory (matsu-www.is.titech.ac.jp) into the replica catalog.

Here, we compare the three replication scenarios:

Intra-Site Replication When the replica of the data to be requested already resides on one of the nodes in the LAN, transfer is made within the node. (1. in Fig 4). Here we artificially register the node that holds the replica into the replica catalog.

Inter-Site Replication from a Remote Server When the replica resides outside the LAN, whereby one of the hosts within the LAN of the group will retrieve the data on behalf of other nodes, and successively performs efficient replication (2. in Figure 4) amongst the nodes.

Parallel Replication with Requests from 2 or more Sites When numerous nodes from multiple groups accesses the same replica residing in the outside server (3. in Figure 4). In this experiment we employed the Dennis Cluster show in Table 3 at University of Tsukuba as the second group accessing the same dataset.

We now present the actual results of the experiment:

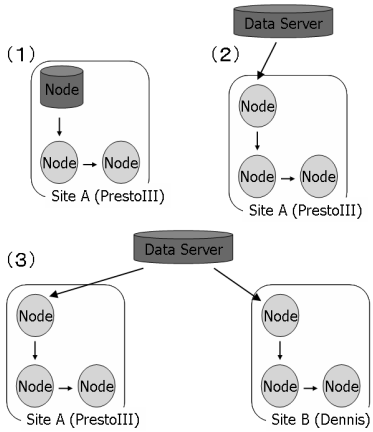

Figure 4. Replication Scenario

CPU	Dual Xeon 2.4GHz
Memory	1GB
Network	1000Base-T
OS	Linux Kernel 2.4.20-28.8smp
Java	Sun JDK 1.4.2_03

Table 3. The Dennis Cluster at Univ. of Tsukuba

4.1. Intra-site replication

Firstly, we compared the Intra-site replication with simple, node-by-node replication using rcp. Figure 5 shows the result. As one can observe, the increase in the replication time is almost negligible, allowing near O(1) replication cost. This is thanks to the efficient and reliable multicast feature of Dolly+, which is being shown to be used to maximum effect.

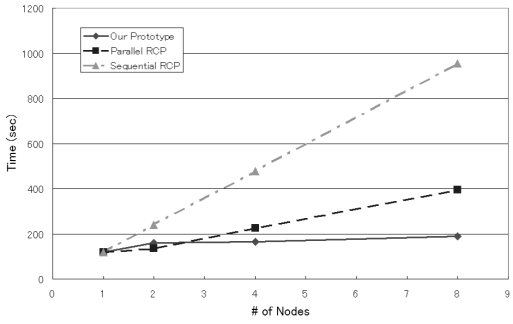

Figure 5. Results of Intra-Site Replication

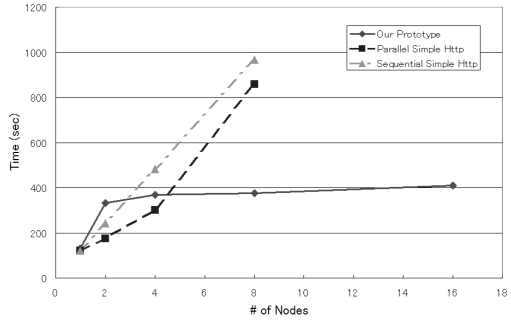

Figure 6. Results of Replication from Remote Data Sever

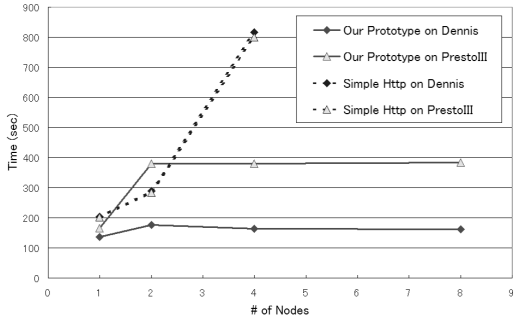

Figure 7. Parallel, Multiple Site Replication

4.2. Inter-Site Replication from Remote Data Server

Secondly, we tested the remote replication scenario whereby a request only comes from a same group of client nodes. For this purpose, we compared the results with the use of more common transfer sequentially or using parallel http transfers.

For brevity, we show the case where the server is close, i.e., a web server in the same lab, This is probably the best case for rcp and parallel http, since the file-by-file transfer bottlenecks are minimized.

Figure 6 shows the results. We observe that, for our prototype implementation, the overhead remains constant even with the increased number of nodes, achieving extremely high scalability. For smaller number of nodes, however, parallel http wins; this is primarily due to the overheads present in our system, which we hope to minimize in the future.

4.3. Parallel, Multiple Site Replication

Finally, we evaluated the scenario whereby two groups of different sites access the same replica, from Tokyo and Tsukuba, respectively. Here we compared the results with parallel http transfer. Figure 7 shows the results. As was with the previous scenario, our prototype exhibits high scalability, at the cost of some overhead for small number of nodes. We also note that the Dennis cluster exhibits superior performance due to its LAN being 1GigE as opposed to 100Base-T on Presto III, indicating that Dolly+ is able to exploit the availability of high local bandwidth. Contrastingly, for http transfers, the results for the two clusters are largely identical. This is because the overhead in the transfer is being dominated by the bottleneck accessing the remote data server.

5. Conclusion

We proposed a parallel, multi-site replication framework for Data Grid that couples data location service in a replica catalog dynamically with efficient and reliable O(1) application-level multicasting. Although the framework is rather simple, we have demonstrated through our prototype implementation that the scheme demonstrates extremely high scalability even in cases where nodes in the hundreds will access the same data set almost simultaneously, compared to traditional replication systems that largely relied on point-to-point protocols.

For future work, we hope to improve our algorithm by fully supporting multicasting over wide-area networks, as well as experimenting with various aggregation algorithms to investigate what would be the most appropriate for more realistic applications. Another issue is exploiting more information from the monitoring services for replica selection, and finally, we hope to build a more solid version which we plan to incorporated into our Data Grid systems such as the Grid Datafarm, as well as other systems, allowing testing in much higher numbers in terms of its scalability.

Acknowledgements

We would like to thank the Grid Datafarm (GFarm) AIST/KEK/U-Tokyo/Titech team for valuable insights regarding replication in the data grid problem space, especially Atsushi Manabe @ KEK who made the original improvements to the Dolly system to create Dolly+, and continued to work with us in improving Dolly+ over the past several years. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas, 13224034, 2004.

References

[1] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, pages 23:187–200, 2001.

[2] An L. Chervenak, Naveen Palavalli, Shishir Bharathi, Carl Kesselman, and Robert Schwartzkopf. Performance and scalability of a replica location service. *The Thirteenth IEEE International Symposium on High-Performance Distributed Computing*, 2004.

[3] W. Allcock, J. Bresnaham, I. Foster, L. Liming, J. Link, and P. Plaszczac. Gridftp update january 2002. *Globus Project Technical Report*, 2002.

[4] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, and Satoshi Sekiguchi. Grid datafarm architecture for petascale data intensive computing. *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 102–110, 2002.

[5] Blast. http://www.ncbi.nlm.nih.gov/BLAST/.

[6] Atsushi Manabe. Disk cloning program 'dolly+' for system management of pc linux cluster. *Computing in High Energy Physics and Nuclear Physics*, 2001.

[7] Apache xindice. http://xml.apache.org/xindice/.