Striped Replication from Multiple Sites in the Grid Environment*

Marek Ciglan, Ondrej Habala, and Ladislav Hluchy

Institute of Informatics, Slovak Academy of Science,
Dubravska cesta 9, 845 07 Bratislava, Slovakia
{marek.ciglan, ondrej.habala, hluchy.ui}@savba.sk

Abstract. Grid technology, as a highly distributed computing environment, requires an optimized access to the data resources to increase data availability. In this paper, we propose a replication technique which is based on parallel transfers from multiple sites containing replicas of the desired file. From each site, we transport in parallel only a portion of the given data source, obtaining the whole file at the end of the process. We describe the work related to the data replication; then we discuss two algorithms for striped replication optimization that aims at the minimization of the time necessary for data transfer. Finally, we present the results of the striped replication mechanism achieved by the prototype implementation of the striped replication algorithm. We compare them with the results of the standard replication tools and show interesting performance improvement.

1 Introduction

Grid computing is an important new concept for distributed processing, based on the idea of globally shared computer resources between organizations, such as disk space, information and computational power [8]. Grid computing helps users, who need to run computational and data intensive tasks, which could be too demanding and time consuming for a single supercomputer or computational cluster.

Such a task is distributed within the grid to several grid nodes, saving execution time. This brings immense computational power for relatively small cost. In such highly distributed environment, an efficient data management is needed to keep track of the data sources in the grid and to optimize the network traffic. Many grid applications request large data collections, which, possibly, have to be transferred via network to the grid nodes that execute given jobs. To minimize the network traffic, transportation of the files, the data replication strategy

* This work is supported by EU 6FP RI(III) project: Enabling Grids for E-sciencE (2004-2006) INFSO-RI-508833, EU 5FP IST RTD project: CROSSGRID Development of Grid Environment for Interactive Applications (2002-05) IST-2001-32243 and the Slovak Scientific Grant Agency within Research Project No. 2/3132/23.

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 778–785, 2005.

is used. Data replication means creating copies, replicas, of files that are often required by grid jobs. This leads to having multiple copies of a single file across several grid nodes, which helps to reduce access latencies to the data.

In general, there are two levels of data access optimization: short-term and long-term optimization. Short-term optimization aims at delivering replica of required file in the shortest time possible, given a requesting grid node and logical file name.

Long-term optimization concerns global reduction of network traffic in the grid by deciding which files should be replicated (deleted) on (from) which grid sites. Current state-of-the-art grid data management tools select the file replica with lowest access cost, given a requesting grid node and logical file name of the data source. The selected replica is then transferred to the grid site. We describe work related to data replication in section 2. In this paper, we present an approach to the short-term replica optimization for grids, that has the potential of further acceleration of the replication process, in the case when more than one replica of desired data source is present in the grid. We propose a striped replication mechanism, a method which doesn't transfer the data from a single grid site only, but is rather based on parallel transfers from multiple sites containing replicas of the desired file. From each site, we transport only a portion of the given data source, obtaining the whole file at the end of the process. Doing so in a parallel way, we can expect time reduction of replica creation. However, transferring file stripes from multiple sites can eventually increase the transfer time, if the process isn't supervised and optimized. For example, if one of the file replica resides on the grid site with extremely low connectivity, attempt to transfer the identical potions of file from each site could results in decrease of overall replication speed. Optimization procedure for striped replication management is thus needed. We discuss two optimization algorithms for the striped replication in the grid environment in section 3.

First algorithm computes portions of replicas that will be transferred from distinct grid sites, according to the information about replica's access costs obtained from monitoring services. This is done before the transfer actually begins, the method is static during the transfer period. Second algorithm is more flexible during runtime and can dynamically change the amount of data that is transferred from different sites, according to actual network performance. We outline the reasons why we choose the second algorithm for the prototype implementation of the striped replication tool.

In section 4, we present the results of replication tests obtained by our prototype implementation of striped replication. We compare them to the results obtained by standard replication tools and show interesting performance increase. Finally, we discuss the advantages and disadvantages of our approach.

2 Related Work

A short-term replica optimization service (ROS) for grids was implemented in European Data Grid (EDG) project [3]. Main function of EDG ROS service is

to evaluate a cost for accessing a set of files from grid sites and to choose the cheapest replica to transfer. It uses gridFTP functionality to increase transfer speed by opening multiple TCP streams from a single site.

A lot of attention to the acceleration of the file replication process was given in peer-to-peer (p2p) systems such as BitTorent, Slurpie, Gnutella.

BitTorrent [9] is a file distribution system. Shared files, within this framework, are made available using HTTP server. When multiple users are downloading the same file at the same time, they upload the pieces of the file to each other. System keeps metadata information about each file, containing file name, size, hashing information and url of a tracker. Trackers are services that help file downloaders to find each other. Information from the tracker is used to find other peer and download management is then handled strictly in the interaction between peers. Shared files are logically split into pieces, downloading peers propagate information about file pieces they have available to each other. The 'rarest first' method is used to decide which file's block would be transferred. BitTorent clients download the rarest file's pieces first, leaving more common ones for later. To prevent transfer slow-down, because of slow transfer rates from certain peers, a transfer choking feature was introduced. Peers try to maximize its own download rates by transferring from whoever they can and deciding which peers to upload file pieces. The choking is a temporally refusal to upload file to the given peer. Peer A stops sending blocks to peer B, chokes the connection to B, until B sends A a block or a time out occurs. The choking encourages peers cooperation. Decisions as to which peer to upload and download from are based strictly on current download rate. As calculating current transfer rate meaningfully is a difficult problem (the bandwidth shifts rapidly over time), BitTorent peer change a single connection, performs 'optimistic unchoke', regardless of the current download rate.

Slurpie [10] is a p2p protocol for bulk data transfer, specifically designed to reduce client download times for large files and to reduce load on servers. All nodes downloading the same file contact the topology server and form a random mesh. The nodes in the random mesh propagate progress updates to each other. This information is used to coordinate the data transfer. Slurpie uses an bandwidth estimation technique to make an informed decision about number of connections to keep open, number of edges to keep in mesh and update propagation rate. Considering the download decision, Sulrpie download blocks served by peers before block served by the source server. Sulrpie adapts to varying bandwidth conditions and scale the number of neighbors as the group size increase.

Both p2p and grid systems use replication strategy to increase data availability, however the setting of p2p and grid environments are different. Connections and disconnections of nodes (users) in the p2p system are quite dynamic, grate effort is focused on locating required file's replicas and on discovering which peer has which file blocks. In the current state of grid environment, we can easily determinate locations of file replicas and only complete replicas, containing all blocks, are present. We try to introduce striped replication principle in the scope of available grid services and we focus primary on the problem of data transfer optimization.

3 Striped Replication from Multiple Grid Sites

The striped replication in grid environment is enabled by capability of GridFTP protocol, which allows the access and the transport of only a portion of the whole file from a grid site (site's Storage Element).

The idea behind striped replication is to transfer portions of desired file's replicas from different grid nodes to further increase the speed of replication. Theoretically, if there are two replicas of the given file in the grid, placed on the sites with the same connectivity considering the site where we want to replicate the file and we replicate this data source in the striped way, 50% of the file from one site and complementary 50% of the file from other site, we can achieve 50% acceleration of the transfer comparing to the replication from a single source. The attempt to transfer file in a striped way may potentially result in decreasing the file transportation speed, if the process is not managed properly according to the actual state of the grid system. For example, if one of the file replica resides on the grid site with extremely low connectivity, attempt to transfer the identical potions of file from each site could results in decrease of overall replication speed. In this section, we describe two algorithms for optimization of the striped replication and we compare them.

The first one is a static approach which computes the portions of distinct replicas that will be transferred before the transfer begins. This method use information from optimization service about the transfer speed from involved grid sites. The second algorithm is more dynamic in nature, it starts by trying to transfer even portions of replicas from different sites and in the course of the data transmission it changes dynamically the amount of data for transfer from different sites, according to the actual performance of the process.

3.1 Static Striped Transfer Mechanism

Our task is to replicate a file in a striped way from multiple grid sites and to do this by computing the portions of involved replicas which will be transferred, before replication actually begins. We can formulate this problem in the following way: Let's have replicas $r_1, \ldots, r_n$ of the data source with given GUID. For each r_i, there is a constant b_i which expresses the amount of bytes that can be obtained from given replica per second. The constants b_i are acquired from the replica optimization service and define current connectivity to the replica. Using this information, we want to find the optimal distribution of replicas partitions for distinct r_i, so that the time of the replication is minimized. We use the following equation as a basis of our thoughts:

$$s = t(b_1 + b_2 + \ldots + b_n)$$

Where s is the given file's size, $b_1, \ldots, b_n$ are the byte-per-second constant for each replica and t is the transfer time. This is a simplified and naive approach because:

- Numbers b_i are considered as constants for the single file replication, which doesn't reflect the dynamic, changing nature of network load.

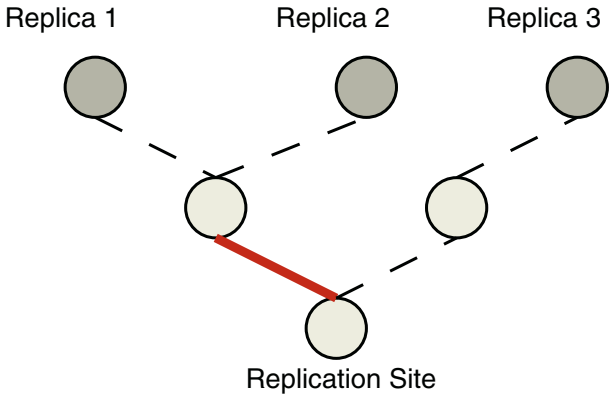

Fig. 1. Shared transfer line for Replica 1 and Replica 2 on the path to Replication Site

- We do not know the exact topology of the network. So if transfer paths from r_i and r_j to the replication site, shares at some point the same line, the transfer speeds b_i, b_j may be eventually lower in reality than those rendered by optimization service, because of available bandwidth of the shared line (figure 1)
- The sum of all b_i can eventually be higher than available bandwidth at local site

More reasons could be found to confirm the naivety of presented approach, still we can use it as a rough approximation. From equation above we can compute the minimal time needed to transfer the file. Now we can simply obtain desired amount of replica portions for all r_i as $p_{r_i} = b_i t$. Doing so, we gather sufficient information to begin the striped replication process.

3.2 Dynamic Striped Transfer Mechanism

In this subsection, we propose the dynamic striped replication mechanism. This method doesn't use the information from the replica optimization service about the connectivity of sites which stores replicas of the desired data source. The only information we have at the beginning of the process, is the location of those replicas.

We begin the striped replication process by initiating n striped transfer threads, each for distinct replica and we try to download even portions of data by all the initiated transfer threads. In the ideal case, this will happen and parallel transfers from replicas grid sites will finish at the same time and the replication time will be $\frac{1}{n}$ of the time required for the replication from a single site.

However, we cannot expect this to happen really often. We propose an optimization technique for dynamic change of the transferring file portions. When some of the parallel striped transfer thread finishes it's replica portion download

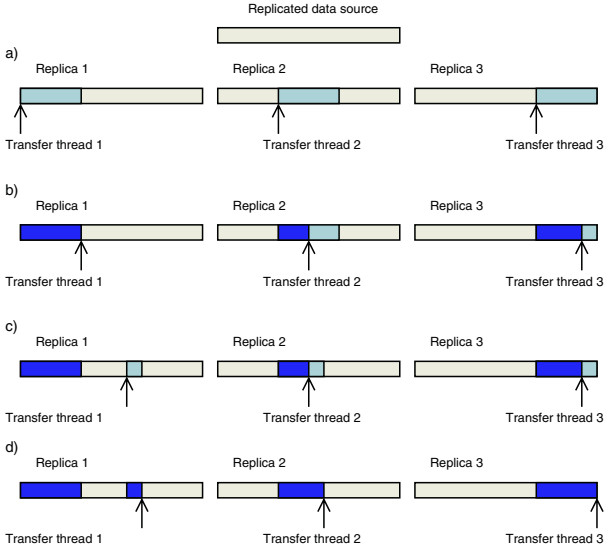

Fig. 2. Illustration of striped replica transfer. Light blue parts symbolize file portions assigned to transfer, dark blue parts sybolize portions already transfered.

a) Situation after the initiation of striped transfer
b) Situation after one of the transfer threads finishes it's assigned portion(s) of replica
c) Situation after reasigning replica portions to transfer threads
d) Possible situation at the end of the process

(and would be idle for the rest of the execution if not handled otherwise), we assign to this thread another portion of the replica, which is equal to the half of the unfinished portion of the transfer thread with largest unfinished amount of data and remove equivalent portion from the latter.

In other words, let there be n replicas of a data source which we need to replicate. We initiate n transfer threads for distinct stripes of the file at different n grid sites and begin transfer. Suppose that thread tr_i finishes transfer of assigned replica stripe and there are other threads that are still transmitting their portions of replicas. Let tr_j be the thread which had transferred the smallest amount of the data. Suppose that tr_j was initiated to replicate portion of the file from the byte p to the byte r and till present, k bytes was downloaded. We split the interval $< p + k, r >$ to two halfs and assign $< p + k, (r\frac{p+k}{2}) >$ to tr_j and $[(r\frac{p+k}{2}), r >$ to thread tr_i.

This process is repeated each time some striped transfer thread finishes it's assigned portion(s) of replica. For better illustration of the process see figure 2. Dynamic striped replication from multiple grid sites reflects the changing status of network load and so, brings necessary flexibility to the replication process. Moreover, this method doesn't need to use information from the replica optimization service.

4 Experimental Results

Intuitively, striped parallel transfer brings important increase of replication speed. The acceleration of replication process clearly depends on the data transport speed from involved replica sites. To give reader the idea of such replication method performance in the real grid environment, we present in this section our experimental results.

We have implemented the prototype of striped replication from multiple grid sites and tested its performance in grid environment. The implementation is made in Java programing language with use of CoG 1.2 API libraries - developed by Globus Alliance [6]. The dynamic striped replication strategy was used in the implementation. We used EDG replica manager tool as a reference implementation of grid data replication, to compare obtained results.

Before presenting averages of the replication time savings during tests, we describe one motivation test case. We replicated a file of 223.9Mb size [1]. There were two replicas of given data source in the testbed. Transfer time of the best replica to the local node using EDG replica manager tool took 713 seconds, using one tcp stream from the site which stored the cheapest replica (the replica with the lowest access cost with the respect to the replication site). The transfer time of striped replication from both gird sites containing given file's replica took 405 seconds (using 1 tcp stream from each site), performing 43% time saving for data transfer. Then we replicated the file to third grid site and run the striped replication from three sites. This transfer took 209 seconds, accomplishing 71% time saving comparing to previously done EDG replica manager transfer. We performed test with two and three replicas of the replicated data source in the grid, obtaining in average 37% time saving for two and 55% for three replicas (compared to the transfer time of the replica from a single site which was selected as the cheapest by EDG replica manager tool).

Concerning disadvantages, striped replication isn't very useful for small data sets. Another disadvantage is that, as the file is transferred in smaller portions, data stripes have to be joined to the single file after the transfer is completed. This requires additional time at the local site.

The advantages of striped replication approach are:

- important acceleration of replication process
- distribution of network load
- method doesn't use the replica optimization service, monitoring services.

5 Future Work

We plan to integrate described mechanism with Replica Location Service, to obtain list of data source's replicas automatically (providing only data source's

[1] The replicated file was actually a short video presenting Data Grid project; we used it as a kind of tribute to the fine work done in this project.

LFN or GUID) and to implement striped replication from multiple grid sites as a web service.

6 Summary

In this paper we have presented the method of striped replication from multiple sources in the grid environment. We have proposed two optimization algorithms for the striped replication. Finally we have presented experimental results of the tests of the prototype implementation of striped replication, performed in the grid environment. The results of tests are promising, they show important time savings compared to currently used method.

References

1. Chervenak A., Deelman E., Foster I., Guy L., Hoschek W., Iamnitchi A., Kesselman C., Kunszt P., Ripeanu M., Schwartzkopf B., Stocking H., Stockinger K., Tierney B., Giggle: A Framework for Constructing Scalable Replica Location Services, Proceedings of SC2002 Conference, November 2002.
2. The Globus Project, www.globus.org
3. EU Data Grid project, WP2, replication, RLS, http://edg-wp2.web.cern.ch/edg-wp2/replication/
4. EU Data Grid project, WP2, optimization, ROS, http://edg-wp2.web.cern.ch/edg-wp2/optimization/ros.html
5. Kunszt P., Laure E., Stockinger H., Stockinger K., Advanced replica management with Reptor, 5th international conference on parallel processing and applied mathemetics, 2003
6. Commodity Grid Kits, http://www-unix.globus.org/cog/
7. Manohar M., Chervenak A., Clifford B., Kesselmann C., A Replica Location Grid Service Implementation, GGF 10 Workshop, 2004
8. I. Foster and C. Kesselman. Computational Grids. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, 1999
9. Cohen B., Incentives Build Robustness in BitTorrent, http://bittorrent.com
10. Sherwood R., Braud R., Bhattacharjee B., Slurpie: A Cooperative Bulk Data Transfer Protocol, Proceedings of IEEE INFOCOM, March 2004