A Hierarchical Scheduling and Replication Strategy

A. Horri, R. Sepahvand, Gh. Dastghaibyfard
Department of Computer Science & Engineering,
College of Engineering, Shiraz University,
Molla Sadra Ave, Shiraz, Iran 71348-51154

Summary
In data grids huge amount of data are generated and processed by users around the world. Objective of dynamic replica strategies is reducing file access time by reducing network traffic, which leads to reducing job runtime. In this paper a replication algorithm for a 3-level hierarchical structure and a scheduling algorithm are proposed. The simulation results with optorsim show better performance (over 13%) comparing to current algorithms.

Key words:
Data replication; Data Grid, Job Scheduling, Simulation

1. Introduction

Grid was first proposed by Foster and Kesselman [1], which is a mechanism for resource sharing and problem solving in heterogeneous environment. Grid is a wide heterogeneous distributed system with multiple administrative domains for application that needs huge amount of computational and storage resources.

An important kind of grid is data grid, which is used in data-intensive applications such as, High Energy Physics (HEP), Genetic, Earth Observation, …. In such applications, the sizes of the data reach tera byte or even peta byte. In this situation, managing and storing this huge data is very difficult or even impossible [1]. Data grid tries to store this data in decentralize sites and then for each application retrieves it from these sites.

The major obstacles that degrade performance of data grid are latency of wide area networks and the bandwidth of internet. Therefore, latency of network and bandwidth between storage sites and processing sites plays an important role in data grid.

In order to increase performance of data grids one needs to consider the followings:

1- Deciding where to allocate the job with respect to location of replicas and computational capabilities of the sites i.e. scheduling optimization.

2- Deciding from where to fetch replicas by considering available network bandwidth between sites i.e. short-term optimization.

3- Deciding what replica files to keep or delete when there is shortage of storage in a site i.e. long-term optimization or dynamic replication strategy.

Replication can be static or dynamic. In static replication the replicas have to be manually created, deleted and managed and it becomes tedious with the increase in files and user jobs. So it has the drawback that can not adapt to changes in user behavior. In real scenario, where the data amounts to peta bytes, and the user community is in the order of thousands around the world; static replication does not sound to be feasible.

Dynamic replication strategies overcome the problem, where replica creation, deletion and management are done automatically. Dynamic replication strategies have the ability to adapt to changes in user behavior. Dynamic strategies are explained in section-2. In replication consistency is an important issue that needs to be considered. To overcome this problem as in other papers, it is assumed that: access pattern is read only for all replicas in data grid.

The remainder of paper is organized as follows. Related work on replication and scheduling is given in section 3. A 3-layerd hierarchical structure is proposed for replication in data grid based on classification of networks, along with a novel algorithm for this structure is given in section 4. Section 5, covers the simulation result with optorsim and section 6 concludes the paper.

2. Replication and scheduling strategies

In this section some of the replication and scheduling strategies are explained as well as file access patterns.

Dynamic replication strategies are as follows:

No replication: do not replicate any file.

Best client: select the best client for replication based on the maximum request for specific file.

Plain caching: the site that requested files stores a copy of them locally.

Fast spread: replica created on the path to the best client.

Dynamic scheduling strategies are as follows:

Manuscript received August 5, 2008.
Manuscript revised August 20, 2008.

Random: jobs randomly distributed to sites.

Job data present: schedule job to a site that has most of the required files, so there will be minimum demand to replica.

Job least loaded: schedule job to a site that has the minimum queue length.

Job locality: schedule job according to local sites that means by grid structure.

And finally file access pattern in a job are as follows:

Random: job requests files with random distribution.

Locally: job requests local files first.

Geographical: job requests both local and non local files.

Selecting appropriate strategy and access pattern in data grid absolutely depends on running applications.

3. Related work

In this section recent works will be considered. In [2], an algorithm for a 2-level hierarchical structure based on internet hierarchy (BHR) has been introduced which only considers dynamic replication and does not consider scheduling. Nodes in the first level are connected to each other with high speed networks and in the second level via internet. The algorithm replicates the file to the site if there is enough space. Next it, accesses the file remotely if the file is available in the sites that are in the same region. Otherwise it tries to make available space by deleting files using LRU (Least Recently Used) method, and replicates the file. It assumes that master site always has a safe copy of file before deleting.

In [3], a structure with few networks connected via internet has been presented and an algorithm similar to [2], along with scheduling is proposed. For replicating a file, first computes the total transfer time, then it selects the best node with shortest transfer time.

In [4] they introduce dynamic replication placement (RP) that categorizes the data based on their property. This category is used for job scheduling and replication. Then a job is allocated to a site which has the file in the required category. This leads to reduce the cost for file transfer.

In [10], it considers a hybrid of tree structure with ring topology as its grid environment. Tree structure provides scalability and ring topology makes it possible to access the data with low latency. Nodes that are physically near each other are connected with high bandwidth and grouped together. They proposed an algorithm that tries to optimize, access rate to data, traffic rate, read cost and write cost parameters for each file.

In [11], it considers two centralized and decentralized replication algorithms. In centralized method, replica master uses a table that ranks each file access in descending order. If a file access is less than the average, it will be removed from the table. Then it pop files from top and replicates using a response-time oriented replica placement algorithm. In the decentralized method, every site records file access in its table and exchange this table with neighbors. Since every domain knows average number of access for each file and then deletes those files whose access is less than the average, and replicates other files in its local storage.

4. The proposed method

In this section, first we present the network structure that is considered in this paper, and then two algorithms are proposed, one for replication and the other for scheduling.

4.1 Network Structure

It is better consider a real scenario that is very frequent in most academic and research centers. In an academic center there are several schools (or faculties) that are usually dispersed from each other, we call them regions. Within each school (region) there are several departments, where each department has its own local LAN. Computers within each local LAN, constitute grid nodes. A schematic structure (Grid structure) of such a network is depicted in figure 1. This network has hierarchical structure, and has three-levels.

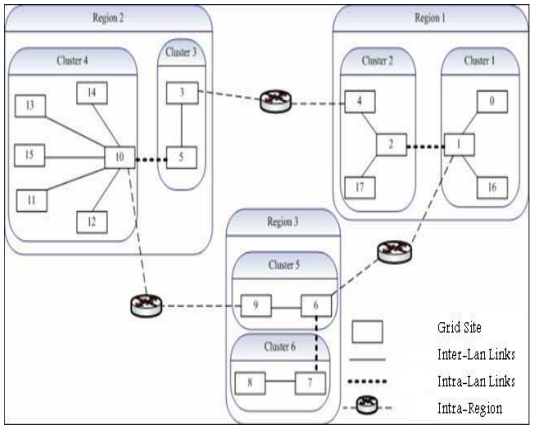

Figure (1) Grid structure in an academic center

Regions comprise the first level. Due to the far distance between the regions, they are connected via internet which has low bandwidth. Regions may also represent cities of a country, or to a larger scale they may represent the countries within each continent with no direct link between them. Next level contains local LANs within the

region, which they have a moderately higher bandwidth comparing to internet between them. Third level comprises the computers within each local LAN, which are connected by a high bandwidth. These nodes have computing element (CE), storage element (SE), or a combination of both.

4.2 The proposed replication algorithm

The algorithm first checks replica feasibility. If it is not feasible (i.e. the requested file size is greater than SE size), file will be accessed remotely. Next it prepares a list of candidate files for replication. From this list, it chooses a file that has the highest bandwidth to the requester node. If the available space in SE is greater or equal to requested file size, it replicates the file. If not, it checks, if the requested file is available in the local LAN, then the file will be accessed remotely. Otherwise, SE is already filled up, and some files should be deleted based on the following steps:

Step 1: Among the files that are available in the file site, prepare a list of candidate files for deletions that are also available in the local LAN, i.e. a copy of them are available in the local LAN in case it is needed later. Sort this list using LRU method. Start deleting the files from this list till it has enough room for the requested file.

Step 2: If deleting all the files in Step 1, does not help, repeat step 1 for each available LAN in the current region randomly, till there is enough room for the requested file.

Step 3: If deleting all the files in Step 1 and 2, does not help, then sort the remaining files by using LRU method and start deleting the files from this list till it has enough room for the requested file.

The complete replica algorithm is given in figure 2.

```
if (the requested file is not available in the site ) {
  if ( requested file size ≥ SE storage size) //  Access file remotely
            Access file remotely ; exit
  else
    if ( requested file size ≤ SE available storage size) //  SE has some free space
      replicate the file; exit;
    else          // SE is filled up
      if ( requested file is available in the local LAN) //  Access file remotely
            Access file remotely ; exit
      else{//SE is filled up and requested file is not available in the local LAN,
          // delete some files that a copy of them is available in local LAN
          // or the LANs in region

        FileList = all data-files in local storage that are also available in the local LAN
        Sort FileList using LRU
        While (FileList!=Empty) {
          select a file from  top of  FileList
          delete this file ;
          if ( requested file size ≤ SE available storage size) copy the file and exit;
        } // endwhile

        foreach (LAN in the region randomly) {
          FileList = all data-files in local storage that are also available in the local region
          Sort FileList using LRU
          While (FileList!=Empty) {
            select a file from  top of  FileList
            delete this file ;
            if ( requested file size ≤ SE available storage size)
                copy the requested file and exit;
          } // endwhile
        } // end foreach

        FileList = remaining data-files in local storage
        Sort FileList using LRU
        While (FileList!=Empty) {
          select a file from  top of  FileList
          delete this file ;
          if ( requested file size ≤ SE available storage size)
                copy the requested file and exit;
        } // endwhile
}
```

Figure (2) Proposed replica algorithm

4.3 The proposed hierarchical scheduling algorithm

For efficient scheduling of any job, the algorithm determines most appropriate region, LAN and site respectively. An appropriate region (LAN, site) is a region that holds most of the requested files (from size point of view). i.e. most of the requested files are available in that region. This will significantly reduce total transfer time, and consequently the network traffic. In other words let:

$S_{r,l,s}$ be total size of data the requested files available in site *s*, LAN *l*, and region *r*.

$L_{r,l}$ be total size of the requested data files available in LAN *l*, region *r*, i.e. $L_{r,l} = \bigcup_{j=1}^{k} S_{r,l,j}$, k is the number of sites in LAN *l*.

R_r be total size of the requested data files available in region *r*, i.e. $R_r = \bigcup_{j=1}^{p} L_{r,j}$ where, p is the number of LANS in region *r*.

Now the algorithm can be summarized as follow:

1- Compute R_r for each region

2- Select the appropriate region $R_{\max} = \underset{j=1}{\overset{q}{Max}} R_j$, q is the number of regions, i.e. region with largest available requested data files from size point of view.

3- for each LAN in $R_{\max}$ **:**

3.1 Select the appropriate LAN, i.e. $L_{max,max} = \underset{j=1}{\overset{t}{Max}} L_{max,j}$, where t is the number of LANS in region R_{max}, i.e. LAN with largest available requested data files in region R_{max},

3.2 for each site in $L_{max,max}$ **:**

3.2.1 Select the appropriate site $S_{max,max,max} = \underset{j=1}{\overset{u}{Max}} S_{max,max,j}$, u is the number of sites in LAN $L_{max,max}$, region R_{max}, i.e. site with largest available requested data files in LAN $L_{max,max}$, region R_{max},

3.2.2 Schedule the job for executing in site $S_{max,max,max}$, LAN $L_{max,max}$, region R_{max}.

To give an example how the scheduling algorithm works, assume we have the following network based on the network structure given in section 4.1:

$R_1=\{L_{1,1}(S_{1,1,1}(f_1,f_2,f_3), S_{1,1,2}(f_2,f_4,f_5), S_{1,1,3}(f_1,f_4,f_6)), L_{1,2}(S_{1,2,1}(f_2,f_4,f_7), S_{1,2,2}(f_6,f_7,f_8))\}$

$R_2=\{L_{2,1}(S_{2,1,1}(f_1,f_2,f_4), S_{2,1,2}(f_3,f_5,f_9)), L_{2,2}(S_{2,2,1}(f_1,f_3,f_6), S_{2,2,2}(f_7,f_8,f_{10}))\}$

f_i means file i is available in that site. For simplicity, assume size of files are all the same (i.e. C=100 MB) and the bandwidth between regions, LANSs and sites are according to table 1 and finally we have a job that requires 5 files (i.e. J={ f_1,f_2,f_3,f_9,f_{10}}) to execute. The proposed scheduling algorithm selects R_2 as the appropriate region and within that region selects LAN $L_{2,1}$ as the appropriate LAN and within LAN $L_{2,1}$, selects site $S_{2,1,1}$ as the appropriate site for executing the above job. In this case files f_3,f_9 should be replicated from site $S_{2,1,2}$ and file f_{10} should be replicated from site $S_{2,2,2}$. So the total file transfer time is (2*C/1000 + C/100)=1.2 seconds.

But if we use simple greedy scheduling algorithms, that selects the site with most available files, it will select site $S_{1,1,1}$ and assign the above job to this site. In this case f_9 should be replicated from site $S_{2,1,2}$ and f_{10} should be replicated from site $S_{2,2,2}$ so the total file transfer time would be (2*C/10)=20 seconds.

5. Simulation

Optorsim is used as the simulation tool to evaluate the performance of the proposed replication and scheduling algorithms. Optorsim was developed to represent the structure of a real European Data Grid [6]. The components of Optorsim are as follow and also depicted in Figure 3.

Resource scheduler: It receives jobs from user and sends them to the best node according to proposed algorithm.

Storage element (SE): Storage resource in grid.

Computing element (CE): Computing resource in grid.

Replica manager: It controls data transferring in each node and provide a mechanism for accessing the Data Catalog.

Replica Optimizer: It has replica algorithm and control file replication according to proposed algorithm.

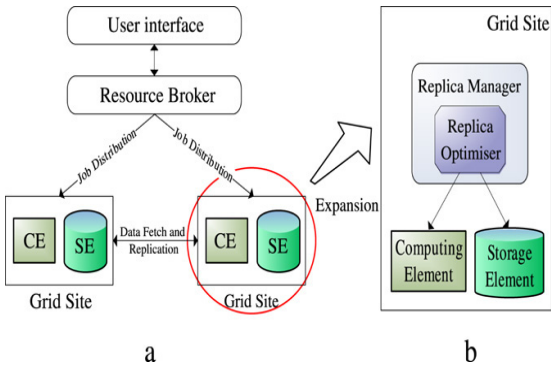

Figure (3) simulator architecture [5]

Based on the scheduling algorithm the broker sends jobs to a node. Each job needs a list of files to run. Reducing file access time is the final objective of optimization algorithms.

5.1 Simulation environment

Since the optorsim structure is flat, modification is done in optorsim code to implement the proposed hierarchical structure.

There are 3 regions in our configuration. Each region has three nodes on average. Table 1 shows the bandwidth configuration (assume **MBS** is Mega Byte per Second) and table 2 shows the job configuration.

Table1 : the bandwidth configuration

Parameter	**Value**
Inter-LAN Link	**1000 MBS**
Intra-LAN Link	**100 MBS**
Intra-Region Link	**10 MBS**

Our configuration has 10 computing elements and 11 storage elements. The average size of storage element is 30 Giga Byte. A node with 150 Giga Byte of storage is used as the main node that holds all initial files. 6 types of jobs have been defined based on access pattern to file. Each job on average needs 15 files (each is 1 giga byte).

Table 2 shows the details. In this simulation as well as in [2, 3, 4], it is assumed that data are read only and valid all the time.

Table 2: job configuration

parameter	Value
Number of jobs types	**6**
Number of file access per jobs	**15**
Size of single file	**1 GB**
Total size of files	**100 GB**

5.2 Simulation Results

The algorithm has been evaluated by comparing it with LRU and BHR. In LRU the files that have not been accessed recently, are candidate for deleting. As these file maybe needed in future and are probably available in other regions, and intra-region bandwidth is low, file transferring time will increase and as a result the job runtime increases.

In BHR algorithm this shortcoming of LRU algorithm has been improved and the probability of deleting files that have only one copy in current region was decreased, therefore the job runtime in comparison with LRU decreased. In BHR algorithm some files which are the only copy inside the LAN, maybe deleted. If these files are needed in future, they should be fetched from other LANs even inside the region and since intra-LAN bandwidth is lower than inter-LAN bandwidth, file transferring time increases and which leads to increase in job run time.

In 3 Level hierarchical structures, this shortcoming has been overcome by considering the differences between intra-LAN and inter-LAN bandwidth and scheduling method.

Figure 4 shows the runtimes of jobs based on changing number of jobs for 3 algorithms. Figure 5 shows the runtimes for 1550 jobs by 3 mentioned algorithms.

If the available storage for replication is not enough, our proposed algorithm will only replicate those files that are not available in the local LAN. So it will not delete those file that have a high cost of transfer i.e. overall low bandwidth between source and destination.

Therefore this method has better performance comparing to LRU. But if the available storage for replication is enough, or there is a high bandwidth between source and destination, both algorithms have close execution time to each other and are similar to LRU.

Overall the simulation results with optorsim show better performance (over 13%) comparing to current algorithms.

Figures 6 and 7 shows the total job time of our and RRU algorithms for varying inter region bandwidth and SE size respectively. As inter region bandwidth and SE size increases both algorithms will converge.

6. Conclusion and future work

In this paper a 3 level hierarchical structure for dynamic replicating file in data grids was proposed. If there is not space for replica, in order to make room, only those file will be deleted that have a low cost of transfer i.e. considering the bandwidth between source and destination. So deletes those file that are available in local LAN. Bandwidth is an important factor for deleting and this leads to a better performance with LRU method. In contrast to BHR algorithm which considers 2-level, the 3 level proposed performs better and it is more realistic.

From job scheduling point of view, the proposed algorithm, first selects the appropriate region (i.e. available maximum requested files), next selects the appropriate LAN in that region and finally selects the appropriate site in that LAN, therefore job execution time decreases since we have minimum data transfer time. Overall the simulation results with optorsim show better performance (over 13%) comparing to current algorithms.

Figure 4 : runtimes of jobs based on varying number of jobs

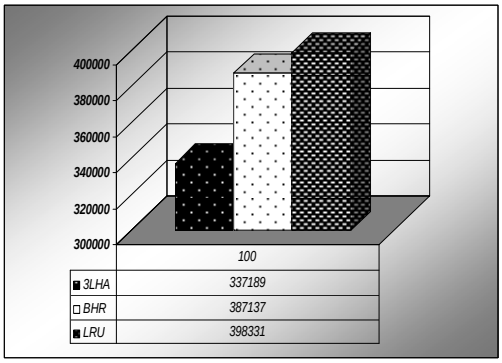

Figure 5: runtimes of the 1550 jobs

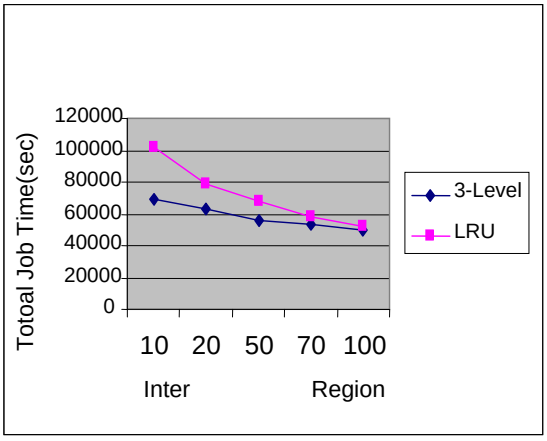

Figure 6: Total job time with varying bandwidth

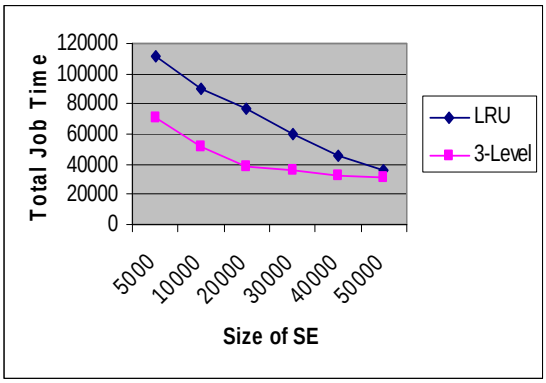

Figure 7: Total job time with varying storage size

References

[1] Ian Foster, Carl Kesselman: The Grid Blueprint for a new computing infrastructure, Morgan Kaufman, 2004.

[2] Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko: Dynamic Grid Replication Strategy based on Internet Hierarchy, Book Series Lecture Notes in Computer Science, Grid and Cooperative Computing book, Publisher Springer, Volume 3033/2004, Pages 838-846.

[3] Ruay-Shiung Chang, Jih-Sheng Chang, Shin-Yi Lin : Job scheduling and data replication on data grids, Future Generation Computer Systems, Volume 23, Issue 7, August 2007, Pages 846-860

[4] Nhan Nguyen Dang, Sang Boem Lim2: Combination of Replication and Scheduling in Data Grids, IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.3, March 2007.

[5] OptorSim - A Replica Optimiser Simulation.

[6] http://grid-data-management.web.cern.ch/grid-data-management/optimisation/optor/

[7] The DataGrid Project. http://www.eu-datagrid.org.

[8] William H. Bell1, David G. Cameron1, Luigi Capozza2,A. Paul Millar1, Kurt Stockinger3, Floriano Zini2 : Simulation of Dynamic Grid Replication Strategies in OptorSim, Book Series Lecture Notes in Computer Science Publisher Springer, Grid Computing — GRID 2002 book, Volume 2536/2002, Pages 46-57

[9] Andrea Domenici, Flavia Donno, Gianni Pucciani, Heinz Stockinger, and Kurt Stockinger : Replica consistency in a Data Grid, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 534, Issues 1-2, 21 November 2004, Pages 24-28.

[10] Houda Lamehamedi, Zujun Shentu, and Boleslaw Szymanski, Simulation of Dynamic Data Replication Strategies in Data Grid, Parallel and Distributed Processing Symposium, 2003. Proceedings. International, Publication Date: 22-26 April 2003, On page(s): 10 pp, ISSN: 1530-2075

[11] Ming Tang, Bu-Sung Lee, Xueyan Tang, Chai-Kiat Yeo - The Impact of Data Replication on Job Scheduling Performance in the Data Grid, Future Generation Computer Systems, Volume 22, Issue 3, February 2006, Pages 254-268