Future Generation Computer Systems 28 (2012) 1045–1057

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

PFRF: An adaptive data replication algorithm based on star-topology data grids

Ming-Chang Lee [a], Fang-Yie Leu [b], Ying-ping Chen [a,*]

[a] Department of Computer Science, National Chiao Tung University, Taiwan

[b] Department of Computer Science, Tunghai University, Taiwan

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form
5 July 2011
Accepted 5 August 2011
Available online 6 November 2011

Keywords:
Data grid
Data replication
Data access patterns
File popularity
PFRF

ABSTRACT

Recently, data replication algorithms have been widely employed in data grids to replicate frequently accessed data to appropriate sites. The purposes are shortening file transmission distance and delivering files from nearby sites to local sites so as to improve data access performance and reduce bandwidth consumption. Some of the algorithms were designed based on unlimited storage. However, they might not be practical in real-world data grids since currently no system has infinite storage. Others were implemented on limited storage environments, but none of them considers data access patterns which reflect the changes of users' interests, and these are important parameters affecting file retrieval efficiency and bandwidth consumption. In this paper, we propose an adaptive data replication algorithm, called the Popular File Replicate First algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space based on aggregated information on previous file accesses. The PFRF periodically calculates file access popularity to track the variation of users' access behaviors, and then replicates popular files to appropriate sites to adapt to the variation. We employ several types of file access behaviors, including Zipf-like, geometric, and uniform distributions, to evaluate PFRF. The simulation results show that PFRF can effectively improve average job turnaround time, bandwidth consumption for data delivery, and data availability as compared with those of the tested algorithms.

1. Introduction

Generally, a data grid, a specific grid system that provides users with a huge amount of storage space, often maintains a high volume of distributed data to serve users. Many recent large-scale scientific systems [1–4] and commercial applications [5], e.g., the Biomedical Informatics Research Network (BIRN) [6], the Large Hadron Collider (LHC) [7], the DataGrid Project (EDG) [8], and physics data grids [9,10], have collected a huge amount of data and performed complex experiments and analyses on the data [11–13].

According to the Pareto principle (also known as the 80/20 rule) [14], a part of data grid files is frequently accessed and transferred. If a file has no replicas distributed over the data grid, the efficiency of accessing the file is often poor since long distance data transfer always occupies a lot of bandwidth and causes long transmission delays [15]. Hence, how to decrease data access latency, lower bandwidth consumption for data transmission, and improve data availability have been the key issues in data grid research [16]. Data replication is a general and simple approach to achieve these goals. It has been widely used in many areas, such as the Internet, peer-to-peer systems, and distributed databases [17–21]. A well-defined data replication method should meet the requirements of being able to determine an appropriate time to replicate files, decide which files should be replicated, and store these replicas in appropriate locations [15,16,22–24].

On the other hand, the analyses of data access patterns have been the critical steps in designing efficient dynamic data replication schemes [25–27]. Several distributions have been used to model data access patterns, defined as the distribution of access counts on files of a system, and file popularity, defined as how often a file is accessed by users, i.e., how popular a file is [28,29]. Breslau et al. [28] claimed that using the Zipf-like distribution can more accurately model the distribution of webpage accesses. Cameron et al. [29] showed that the distribution of file accesses in data grids follows the Zipf-like distribution. Ranganathan and Foster [22,30] claimed that the geometric distribution can properly model file access behaviors and the property of temporal/geographical locality.

Further, Ranganathan and Foster [31] derived file popularity by using both Zipf and geometric distributions on a multi-tier data grid with unlimited storage space. Tang et al. [23] also used Zipf-like and geometric distributions to simulate users' file access behaviors on a multi-tier data grid. Chang et al. [32,24] proposed two data replication strategies on a cluster-based data grid with limited storage. However, the strategy they proposed in [32] did

* Corresponding author.
E-mail addresses: mingchang1109@gmail.com (M.-C. Lee), leufy@thu.edu.tw (F.-Y. Leu), ypchen@nclab.tw, ypchen@cs.nctu.edu.tw (Y.-p. Chen).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.08.015

not consider the data access pattern. Hence, it might lead to inefficient data access as the users' access pattern changes; the strategy proposed in [24] only replicates the file most frequently accessed in the last time period, consequently resulting in long file transmission delays for those files with similar but low weights.

In this study, we propose an adaptive data replication algorithm, called the *P*opular *F*ile *R*eplicate *F*irst algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space. A star-topology data grid is a simplified tree-topology data grid with a central cluster that connects all other clusters. A link l between two arbitrary clusters will go through the central cluster, and l might comprise several routers, and physical links. Directly evaluating the components of l is difficult since too many analytical items might be involved. Hence, this study treats l as a logical link to simplify the original topology as a whole [33, 34]. The simplification process will be proposed. To adapt to the changes of users' interests in files, the PFRF aggregates file access information and replicates popular files to suitable clusters/sites. We simulate several cases in which file popularity follows a Zipf-like distribution, geometric distribution, and uniform distribution under the assumption that user behaviors vary with the changes of users' interests. The simulation results show that PFRF provides users with a system that has higher data availabilities, lower data transmission delays, and less bandwidth consumption for data access.

The rest of the paper is organized as follows. Section 2 introduces background and related work of this study. Section 3 describes the architecture of a star-topology data grid and the details of the PFRF. Simulation results are presented and discussed in Section 4. Section 5 concludes this article and addresses our future research.

2. Background and related work

In this section, we describe the architectures of data grids and several existing replication strategies and algorithms.

2.1. Data grid architecture

Data grids can be classified into multi-tier data grids, first proposed by the MONARC project [35], and cluster data grids, initially introduced by Chang et al. [32]. The multi-tier data grid architecture in which a leaf node represents a user or a computational node, and internal nodes are resource sites keeping sharable files. In this architecture, a file held by a site will also be held by all its ancestor sites. Therefore, the root site holds all files stored in the data grid. When an end user requires a file F which does not exist in his/her site, the user requests F from its immediate ancestor. If the ancestor does not have the file, it in turn requests F from its immediate ancestor. The process repeats until the user obtains the file from a node which holds the file. After that, the file will be replicated to all the nodes on this requesting path following the reverse direction of the requests. It is clear that file access latency can be reduced in a multi-tier data grid, but it leads to higher storage cost since files will be redundantly stored in multiple locations.

A cluster data grid consists of n clusters connected by the Internet [24]. Files are stored in these clusters. Each cluster has a header node (a header for short) responsible for managing site information and exchanging file access information with other cluster headers. A header periodically determines which file should be replicated by computing file weights. After that, the file with the highest weight will be replicated to clusters that need the file. Sites in these clusters can then locally and quickly retrieve the file. Compared with a multi-tier data grid, a cluster data grid consumes less storage to hold files.

2.2. Existing data replication algorithms/strategies

Least Frequently Used (LFU) [36] and Most Frequently Used (MFU) [36] are two simple dynamic replication strategies widely used in many areas, such as disk and cache memory duplication. If a storage device has insufficient space to hold a new file, LFU (MFU) will be invoked to choose the files that have been the least (most) frequently used as the victims to make room for the new one. In the experiments of this study, MFU and LFU are both involved, called the MFU/LFU strategy (M/LFU for short) in which MFU is used to choose the most frequently used files and LFU is employed to select victims once the destination cluster has insufficient storage space to save the replicated files.

Ranganathan and Foster [22] presented six replication/caching strategies for a multi-tier data grid: No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication, and Fast Spread, and three types of localities: temporal locality, geographical locality, and spatial locality. The experimental results showed that the Fast Spread and Cascading Replication outperform the other four strategies and their file access latencies are shorter than those of the other four strategies. They also found that Fast Spread (Cascading) is better when the data access pattern is random (geographical locality). However, the six strategies cannot avoid the disadvantages of a multi-tier data grid, i.e., a file may be redundantly stored in a multi-tier. In fact, the storage space utilization and access latency are a trade-off [32]. Ranganathan and Foster [31] also proposed a suite of job scheduling and data replication algorithms for a multi-tier data grid and evaluated the performance of different combinations of the replication and scheduling strategies. One of the data replication algorithms, called DataRandom (DR for short), replicates a file when the corresponding access frequency exceeds a pre-defined threshold. Although DR is designed for an unlimited storage environment, it can also be run on a limited storage environment. DR is therefore involved in the experiments of this study.

Tang et al. [23] introduced Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) algorithms to reduce the average data access response time for a multi-tier data grid. The basic idea of the two algorithms is to replicate a file to sites close to its requesting clients when the file's access rate is higher than a pre-defined threshold. SBU considers the file access history for individual site, but ABU aggregates the file access history for a system. With ABU, a node sends aggregated historical access records to its upper tiers, and the upper tiers do the same until these records reach the root. Due to the aggregation capability, ABU has a shorter job response time and less bandwidth consumption than those of SBU.

Khanli et al. [37] proposed an algorithm called Predictive Hierarchical Fast Spread (PHFS), which is an extended version of fast spread [22], in a multi-tier data grid. PHFS utilizes spatial locality [22,38] to predict data files required in the future, and pre-replicates these files to suitable sites to improve the performance of file accesses. Kunszt et al. [39] presented a replica management grid middleware to reduce file access/transfer time. Their experimental results showed that this middleware significantly reduces wide area transfer times. However, this model was developed for multi-tier data grids with unlimited storage space.

Chang et al. [24,32] presented two dynamic replication strategies, Latest Access Largest Weight (LALW) [24] and Hierarchical Replication Strategy (HRS) [32], on cluster-based data grids. LALW utilizes the half-life concept to evaluate file weights. A file with a higher access frequency has a larger weight. Their experimental results show that LALW outperforms LFU and no-replication data replication strategies [22] in network utilization and efficiency. However, LALW only replicates the most popular file in each time