Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 24 (2008) 85–98
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

An on-line replication strategy to increase availability in Data Grids☆

Ming Lei*, Susan V. Vrbsky, Xiaoyan Hong

Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487-0290, United States

Received 31 January 2007; received in revised form 16 April 2007; accepted 19 April 2007
Available online 29 April 2007

Abstract

Data is typically replicated in a Data Grid to improve the job response time and data availability. Strategies for data replication in a Data Grid have previously been proposed, but they typically assume unlimited storage for replicas. In this paper, we address the system-wide data availability problem assuming limited replica storage. We describe two new metrics to evaluate the reliability of the system, and propose an on-line optimizer algorithm that can Minimize the Data Missing Rate (MinDmr) in order to maximize the data availability. Based on MinDmr, we develop four optimizers associated with four different file access prediction functions. Simulation results utilizing the OptorSim show our MinDmr strategies achieve better performance overall than other strategies in terms of the goal of data availability using the two new metrics.

Keywords: Data availability; Data Grid; Data missing rate; Limited storage; Replica strategy

1. Introduction

The popularity of data-intensive scientific applications, in which millions of files are generated from scientific experiments and thousands of users world-wide access this data, has resulted in the emergence of Grid computing. In a Grid system, the resources of many computers, spanning geographic locations and organizations, are utilized to solve large-scale problems. These geographically distributed systems with loosely coupled jobs can require the management of an extremely large number of data sets. A Grid computing system for processing and managing this very large amount of distributed data is a Data Grid. Examples of Data Grids are the Biomedical Informatics Research Network (BIRN) [22], the Large Hadron Collider (LHC) [23] at the particle physics center CERN, the DataGrid project (EDG) [21] funded by the European Union, now known as the Enabling Grids for E-SciencE project (EGEE) [5], the International Virtual Observatory Alliance (IVOA) Grid community Research Group [24] and physics Data Grids [6,12]. Data Grids require users to share both data and resources, and the management of such a large volume of data sets has posed a challenging problem in how to make the data more approachable and available to the users.

A common solution to improve availability and file access time in a Data Grid is to replicate the data. When data is replicated, copies of data files are created at many different sites in the Data Grid [3]. Deciding on where and when to make the data copies in the distributed nodes is a problem common to all data replication schemes for Data Grids. Earlier research on data replication [2,4,8,11,14,16] focused on decreasing the data access latency and the network bandwidth assumption. As bandwidth and computing capacity have become relatively cheaper, the data access latency can drop dramatically, and how to improve the data availability and system reliability becomes the new focus.

The dynamic behavior of a Grid user, in combination with the large volume of datasets, makes it difficult to make decisions concerning data replication to meet the system availability goal [15]. In a Data Grid system, there are hundreds of clients across the globe who will submit their job requests, each of which will access multiple files to do some type of analysis. In data-intensive applications, when a job accesses a massive-size file, the unavailability of that file can cause the whole job to hang and the potential delay of the job can be

☆ Earlier versions of this paper were presented at the Third International Workshop on Networks for Grid Applications GridNets 2006 held in San José, California, USA, October 1–2, 2006.

* Corresponding author. Tel.: +1 205 566 4192.
E-mail addresses: mlei@cs.ua.edu (M. Lei), vrbsky@cs.ua.edu (S.V. Vrbsky), hxy@cs.ua.edu (X. Hong).

0167-739X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2007.04.009

unbounded. In large-scale data-intensive systems, hundreds of nodes will be involved and any node failure or network outage can cause potential file unavailability. As a result, there has been an increase in research focusing on how to maximize the file availability. Data replication strategies to improve the data availability have been proposed in [13,15], but have assumed unlimited storage for replicas.

In this paper, we address the system-wide data availability problem assuming limited replica storage. We present two new data availability metrics, the System File Missing Rate and the System Bytes Missing Rate. (We are not aware of any other research at this time that utilizes any system data availability metrics.) We then model the problem in terms of an optimal solution in a static system. More importantly, for on-line processing of file replication, we propose a novel heuristic algorithm that maximizes the data availability by Minimizing the Data Missing Rate (MinDmr) for limited storage resources without sacrificing the data access latency. Based on MinDmr, we present four optimizers that are associated with four different prediction functions. Our test results on the popular OptorSim [9] show that our four MinDmr replica schemes perform better overall than the Binomial economical replica scheme, Zipf economical replica scheme [2,4], LFU and LRU for the two metrics.

The rest of the paper is organized as follows. We describe related work in Grid systems in Section 2. Section 3 presents the two measures (the System File Missing Rate and the System Bytes Missing Rate), and discusses the system model. We present our analytical model and the dynamic replica algorithm in Section 4. In Section 5, we describe our simulation results based on the OptorSim, a simulator designed by the European Data Grid Project [5]. In the final section, we conclude our paper and describe future work.

2. Related work

Work on data availability in Grid systems initially focused on decreasing the data access latency and the network bandwidth assumption. In [14], the six replica strategies (No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread) are simulated for the three user access patterns (random access, small temporal locality, and small geographical and temporal locality). The simulation results show that the best strategy has significant savings in latency and bandwidth consumption if the access patterns contain a moderate amount of geographical locality. In [2,4], a replica scheme based on the economical-model (Eco) has been proposed. The authors use an auction protocol to make the replica decision for long-term optimization. The authors show the scheme outperforms other replica strategies with sequential file access patterns. We note that in Section 5 we compare our proposed strategies to this economical-model.

In [16], Szymaniak et al. present the HotZone algorithm to place replicas in a wide-area network, so that the client-to-replica latency is minimized. They use the GNP [8] technique to model the Internet as an M-dimensional space and all nodes are landmarked into different network regions. HotZone places replicas on nodes that, along with their neighboring nodes, generate the highest load. Sang-Min Park et al. in [11] propose a dynamic replica replication strategy, called HBR, to reduce data access time by avoiding networking congestion in a Data Grid network. The HBR algorithm benefits from 'network-level locality', which indicates that the required file is located at the site which has the broadest bandwidth to the site of the job execution.

Subsequent work focused on maximizing file availability in a Grid system. In their early work, Schintke and Reinefeld present an analytical model in [15] for determining the optimal number of replica servers, catalog servers and catalog sizes to guarantee a given overall reliability in the face of unreliable components. In [13], Ranganathan shows a dynamic model-driven replication approach in which peers create replicas automatically in a decentralized fashion. Both [13] and [15] propose algorithms to meet the data availability goal based on the assumption that the total system replica storage is large enough to hold all the data replica copies. Each file will be replicated to the arbitrary number of copies needed to achieve its availability goal without any discrimination, even if the file will be only be accessed one time in its whole life span. One could argue limited storage resources should not be wasted to hold so many copies of such infrequently used files.

In [17] the authors present a design and implementation of a file-based replica management Grid middleware that was developed within the EDG Project. It was designed so that user communities can adjust the replica behavior based on their quality of service requirements. The goal of this work is to minimize the file access/transfer time. The optimization service gathers latency information from the network and storage element monitoring service to determine which network link should be used to minimize transfer times. Experimental results show their approach significantly reduces wide area transfer times. They do not consider limited storage for replicas.

In [18], the authors study the effect of replication schemes and Grid scheduling heuristics on turnaround time. They assume a set of domains, in which each domain contains a replica server and many computing sites. They consider several replication schemes including centralized dynamic replication and distributed dynamic replication, and several scheduling strategies, such as shortest turnaround, least relative load and data present. Results indicate that dynamic replication algorithms are the most successful in reducing the job turnaround time. The authors do consider limited storage for replicas, but they only consider the LRU algorithm for replica replacement.

Dynamic replication algorithms for multi-tiered Data Grids are presented in [20]. The authors propose two dynamic replica algorithms, Simple Bottom Up and Aggregate Bottom Up, for the multi-tiered Grid. They also develop a Data Grid simulator, called DRepSim. Performance results indicate both of their algorithms can reduce the average response time of data access compared to a static replication strategy in a multi-tiered Grid. The strategies in this work are applicable only to multi-tiered Grids.