สำนักงานวิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER I

Introduction

1.1 Background and Motivation

A grid is a collaborative system built to serve the need for large amount of resources of a big community. Applications in grids can be classified into five major classes -- distributed supercomputing, high-throughput computing, on-demand computing, data-intensive computing, and collaborative computing [1].

First, distributed supercomputing applications use grid to aggregate computational resources, such as CPU and memory, to solve problems that cannot be handled by a single system. One example of distributed supercomputing applications is distributed interactive simulation for training and planning in the military [2]. Second, high-throughput computing uses available processors on grids, such as Condor system at University of Wisconsin [3], for massive jobs. On-demand computing uses remote resources that are not located locally for short-term requirements. For example, NetSolve allows users to couple remote software and resources into desktop applications [4]. Data-intensive applications use data which are geographically distributed on repositories, e.g., digital libraries and databases, on grids to synthesize new information. For example, the digital sky survey uses terabytes of astronomical photographic data which are available on network database for astronomical research [1]. Finally, collaborative-computing applications use grids to support collaborative work between multiple participants such as NICE system, which allows children to create and maintain the realistic virtual worlds for entertainment and education [5]. This thesis focuses on a grid that deals with data-intensive applications which is called a ***data grid***.

Data grids must provide transparent access to data distributed on grids, i.e., users can access data located on any server in a grid as if the data is stored locally [6]. However, it usually takes longer to access a dataset which is stored on other servers because of the transmission delay and processing delay. There are three techniques to reduce the download time -- data replication strategy, replica selection strategy and co-allocation strategy.

Data replication strategy is a process to create or delete replicas based on the need for the dataset [7]. If a client needs a dataset which is not stored locally, the dataset is replicated to the local server or a server in the same area in order to reduce the download time. Due to the limitation of disk storage, all dataset cannot be replicated, and some replicas need to be deleted. Replica selection strategy is used to choose, among all servers with replicas of the required dataset on a grid, the best server to send the replica to the client. The server is chosen based on grid characteristics such as disk access rates and network status [8]. In contrast to data replication strategy, replica selection strategy does not create and delete replicas of data, but selects the best replica from existing ones. However, replica selection strategy chooses only one server to transfer the required dataset. Co-allocation strategy allows parallel data transfer from many servers in order to utilize the bandwidth from all servers. As a result, each server containing a replica of the required dataset sends parts of the replica to the client in parallel. The co-allocator is a process which manages the parallel transfer of data for each request from a client. When the co-allocator gets a request from the client to download a dataset, it assigns each server to send parts of requested dataset to the client. The process is completed when the client receives the whole dataset. The co-allocation strategy is divided into two categories, static and dynamic, based on adaptability criteria. The static co-allocator allocates workload to each server only once before the data transmission starts while the dynamic co-allocator adjusts the workload of each server during the data transmission. Because the dataset is transferred from available servers in parallel, the completion time to transfer the dataset is reduced.

Replicating a dataset on other servers is frequently used in practice to increase data availability as well as download speed. When a dataset is replicated, the whole

dataset is copied to another location. However, it may not be possible to replicate the whole dataset due to some constraints, e.g., storage limitation and data authorization. The concept of fragment replication, which allows replication of some part of data, is proposed [9].

For existing co-allocation strategies, it is assumed that data are completely replicated. When fragments of data are replicated, each fragment may not be of the same size and may not be replicated to the same number of servers. As a result, existing co-allocation strategies do not perform well. Instead of one completed data solved by the existing co-allocation strategies, many fragments of data are replicated over servers in grid. The co-allocator needs to consider more on which fragment should be assigned to reduce the completion time.

1.2 Contributions

This thesis studies how to efficiently transfer a dataset whose fragments are replicated on servers in a data grid. In this thesis, dynamic co-allocation strategy [10] is modified by adding a strategy for choosing fragments of data for each transmission in order to allow fragment replication. When a dataset is completely replicated, dynamic co-allocation strategy can send any part of the dataset at any time. However, when each server does not replicate the whole dataset, choosing which fragment to be transmitted can affect the performance. This thesis studies five algorithms for choosing fragments -- *Random*, *Round-robin*, *Random-with-weighted-probability*, *Biggest-remaining-first* and *Fewest-replicas-first algorithms*.

Random and Round-robin algorithms are used as baselines to compare the performance of other three algorithms. Random and Round-robin algorithms choose each fragment with a certain probability. The difference between Random algorithm and Round-robin algorithm is the probability for choosing each fragment. Random algorithm considers all fragments of a dataset and chooses each fragment with equal probability. On the other hand, Round-robin algorithm considers only fragments

หอสมุดกลาง สำนักงานวิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

replicated in a server and chooses each local replica with the same probability. As a result, a fragment with more replication is chosen with higher probability than one with less replication.

Using Random and Round-robin algorithms as baselines, three other algorithms are studied. Random-with-weighted-probability algorithm considers the original size of fragment as a main factor to select the fragment for each server. That is, a larger fragment is chosen more often than a smaller one. Biggest-remaining-first algorithm chooses the fragment which has largest untransmitted data first. Finally, Fewest-replicas-first algorithm chooses the fragment which is least replicated first. It chooses from fragments replicated on the server which has the fewest replications over the grid first.

The contribution of this thesis is to develop co-allocation strategies for fragmented data. These co-allocation strategies are based on fragment selection algorithms, and allow downloading files with fragmented replicas on data grids.

1.3 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 describes techniques to improve download speed in data grids, which are data replication strategies, replication selection strategies and co-allocation strategies. Then, in Chapter 3, grid architecture for co-allocation is described and fragment selection algorithms to be studied in this thesis are explained. Experiments and results obtained from the simulation of the proposed algorithms are discussed in Chapter 4. Conclusion and future work are presented in Chapter 5.