Multi-Hop Path Splitting and Multi-Pathing Optimizations for Data Transfers over Shared Wide-Area Networks using GridFTP

Gaurav Khanna [1], Umit Catalyurek [2], Tahsin Kurc [2], Rajkumar Kettimuthu [3], P. Sadayappan [1], Joel Saltz [2], Ian Foster [3]

[1]Department of Computer Science and Engineering, The Ohio State University
[2]Department of Biomedical Informatics, The Ohio State University
[3] Mathematics and Computer Science Division, Argonne National Laboratory

ABSTRACT

In this paper, we propose to employ two optimizations – multi-hop path splitting and multi-pathing – to improve the performance of data transfers over shared public networks. We present a path determination algorithm which integrates the aforesaid optimizations in order to improve the performance of single file transfers. Finally, we develop a file transfer scheduling algorithm based on this framework, and evaluate its effectiveness on a wide-area testbed.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Miscellaneous

General Terms

Performance

Keywords

Multi-Hop Path Splitting, Multi-Pathing, Wide-area networks, GridFTP

1. INTRODUCTION

As data continues to be collected and generated at an increasing rate, scientific research is becoming data driven. High-end Grid computing involves use of distributed collections of computational systems to analyze, process and visualize data resident in data repositories. For example, with the Large Haldron Collider (LHC) [1] at CERN, data generated by a CMS [3] experiment must be transferred to the Tier-1 site in USA, where it is processed and then multi-cast onto many domestic Tier-2 sites. Apart from a few high-bandwidth optical network deployments, most of the scientific experiments involve the transfer of data over public, shared network infrastructure. The dynamic nature of shared wide-area networks and the existence of overlay routes to transfer files makes it imperative to design a file transfer mechanism which leverages these characteristics to to maximize the achievable bandwidth.

GridFTP [2] is widely accepted as a secure, reliable and high performance data transfer protocol. It employs TCP as the underlying protocol and creates multiple streams between the source and the destination in order to offset the network congestion. It, however, does not account for the existence of overlay paths.

In this work, we seek to explore the effects of multi-hop path splitting and multi-pathing to improve the file transfer performance in GridFTP. Multi-hop path splitting improves performance by replacing a direct TCP connection between the source and destination by a multi-hop chain through some intermediate nodes. Multi-pathing involves striping the data at the source and sending it across multiple overlay paths. We propose a path determination heuristic which incorporates these optimizations for efficient transfer of a single file. To optimize performance for batch file transfer requests, we extend a collective file-transfer scheduling heuristic implemented in an earlier work [4]. We experimentally evaluate the optimizations and their GridFTP implementation on a wide-area testbed.

HPDC'08, June 23–27, 2008, Boston, Massachusetts, USA.
ACM 978-1-59593-997-5/08/06.

2. DATA TRANSPORT OPTIMIZATIONS

2.1 Multi-Hop Path Splitting

A split-TCP connection involves dividing a TCP connection into a set of shorter connections by splitting it at multiple intermediate points with the goal of improving the overall throughput. It reduces the round-trip time on each intermediate hop as compared to the direct end-to-end path, thereby sensing the maximum TCP throughput on each hop quickly. In this work, as an alternative to a direct TCP connection between a source and destination, we explore the use of multi-hop pipelined transfers using intermediate nodes. If the bandwidth on each of the intermediate hops is higher than the direct path, the overall throughput can be expected to improve.

2.2 Multi-Pathing

Striping the data at the source and sending it across multiple overlay paths can also lead to a better achievable throughput by simultaneously transferring disjoint chunks of a file to its destination. However, seemingly independent overlay paths may share bottlenecks due to physical sharing of links and routers. Therefore, an approach that finds multiple parallel links, but does not consider the physical "underlay" network, will find suboptimal solutions. This is key to maximizing the effective aggregate bandwidth.

2.3 Path Determination Algorithm

The path determination algorithm is an iterative algorithm that computes a set of paths which can be collectively used to transfer a file from its source node(s) to its destination node. At each step, algorithm invokes a variant of the Dijsktra's shortest path algorithm to find a path that will yield minimum transfer time for the requested file. It then modifies the overlay network graph to reflect the current transfer, and continues it search for another path. The weighting function employed for weight assignment to an edge is the the ratio of the round-trip time of the path corresponding to the edge to its bandwidth. The motivation behind this is to give preference to low-latency high-bandwidth edges. The other difference is the calculation of the distance function to measure the goodness of a path. Since the transfer of a file from a source node to a destination node through a multi-hop path occurs in a pipelined fashion, therefore, the distance function of a path is computed as the maximum of the weights on each of its constituent edges. Note that the traditional shortest path algorithm employs the distance function to be the sum of weights instead of the maximum. The proposed algorithm chooses a set of independent paths to collectively transfer a file. However, one or more selected paths can possibly share bottleneck links, which means that the overall bandwidth would not necessarily increase by employing multiple paths. In some cases, the aggregate bandwidth might sometimes even decrease by employing multiple paths. We model the shared bottlenecks by performing an offline characterization of the network and using it to figure out if the two edges share a common bottleneck. This characterization is then used to avoid choosing multiple overlay paths where in the aggregate bandwidth would not increase.

2.4 Scheduling a Batch of File Transfers

In a recent work [4], we proposed a dynamic scheduling algorithm which schedules a set of file transfer requests made by a batch of data-intensive tasks in a wide-area environment. The scheduling algorithm is iterative, employs adaptive replica selection, and makes use of multiple sources for simultaneously transferring multiple pieces of the same file, i.e., non-overlapping portions of a chunk, *sub-chunks*, can be retrieved simultaneously from multiple file replicas. In this paper, we build upon the algorithm proposed in our previous work by incorporating the ideas of multi-hop path splitting and multi-pathing. The scheduling policy is based on MinMin.

3. EXPERIMENTAL RESULTS

We compare our dynamic scheduling approach against our previously proposed work [4] as well as a baseline strategy that we refer to as *Naive Scheduling*. In this approach, each destination site picks a randomly chosen replica source for retrieving a file instead of employing dynamic bandwidth information or multiple replicas. Here onwards, we refer to our previously proposed scheduling algorithm [4] as *GDS* (Global Dynamic Scheduler), the scheduling variant that incorporates path-splitting and multi-pathing optimizations as *GDS-MHMP*, and the scheduling variant that incorporates the modeling of shared bottleneck on top of these two as *GDS-MHMP-SB*. The experiments were carried out across five sites: BMI, a memory/storage cluster at the Ohio State University; ST, the Starlight site in Chicago; JA site in Japan which is a part of the Japan Gigibit Network II (JGN2) project; ORNL, which consists of 28 dual processor 3.06 GHz Intel Xeon sites; and ANL, a IA-32 Linux cluster which consists of 96 dual-processor Intel Xeon sites. Figure 1 shows the performance of the scheduling schemes under a 1-to-all communication pattern. In this experiment, only one of the 5 sites acts as a source initially. Each of the file needs to be transferred to all the other sites. The results show that *GDS-MHMP* leads to significant improvements in the achieved throughput over the *GDS*. This is because it exploits path splitting and multi-pathing. *GDS-MHMP-SB* performs similar or better in comparison to *GDS-MHMP*. The results also show that the proposed schemes are able to consistently outperform the *Naive Scheduling* scheduling approach, as expected.

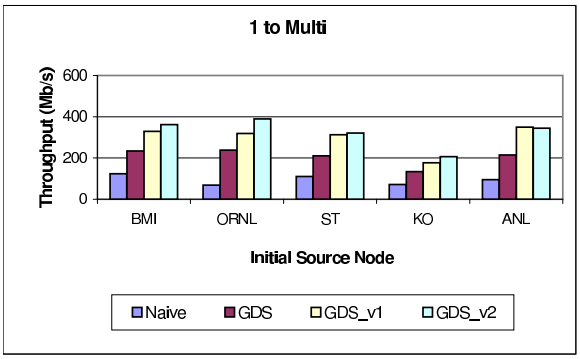

Figure 1: Performance of all the algorithms under the 1-to-all communication pattern in terms of the Average throughput

4. CONCLUSIONS

In this paper, we explored two optimizations, namely, multi-hop path splitting and multi-pathing to improve the performance of file transfers over wide-area networks. We presented a path determination algorithm which integrates the aforesaid optimizations in order to improve the performance of single file transfers. We incorporated these optimizations within our previously proposed wide-area scheduling algorithm and experimentally showed its effectiveness on a wide-area testbed.

5. REFERENCES

[1] The Large Haldron Collider (LHC) . http://lhc.web.cern.ch/lhc/.

[2] W. Allcock. Gridftp: Protocol extensions to ftp for the grid. In *Global Grid ForumGFD-R-P.020*, 2003.

[3] K. Holtman. Cms data grid system overview and requirements. In *Computing in High Energy and Nuclear Physics (CHEP)*, 2001.

[4] G. Khanna, T. Kurc, U. Catalyurek, R. Kettimuthu, P. Sadayappan, and J. Saltz. A dynamic scheduling approach for coordinated wide-area data transfers using gridftp. In *Proc. of 22th International Parallel and Distributed Processing Symposium (IPDPS)*, Miami, Florida, 2008. to appear.