Data Migration in Distributed Repositories for Collaborative Research

Mehmet Balman[1,2], Ismail Akturk[1,3], Somnath Roy[1,4], Tevfik Kosar[1,3], Gabrielle Allen[1,3], Sumanta Acharya[1,4]

1Center for Computation and Technology, Louisiana State University
2Department of Electrical and Computer Engineering, Louisiana State University
3Department of Computer Science, Louisiana State University
4Department of Mechanical Engineering, Louisiana State University

Abstract. In recent years, there has been a growing demand on the required resources in terms of computation and storage to satisfy the needs of scientific experiments and large scale scientific simulations. Scientific applications especially in several areas such as physics, biology, and astronomy have become more complex; and, geographically separated distributed resources have been used to satisfy their immense computational requirements. On the other hand, these applications have become increasingly data intensive such that storage requirements went from giga- to petascale. The distributed nature of the resources made I/O the major bottleneck for end-to-end application performance. Since we need to deal with huge collection of information and distributed data sets, and also these complex applications use heterogeneous resources connected over highly distributed networks, data management middleware has become one of the most important challenges in the area. Therefore, we present a data placement scheduler, for mitigating the data bottleneck in collaborative peta-scale computing systems.

Key words: Data Migration, Transfer Optimization, Collaborative Research, CyberTools, PetaShare

1 Introduction

Data management has perpetually remained one of the crucial problems in every stage of computer engineering, from micro (CPU chip design) level to macro (Internet and Grid infrastructure) level [1]. For example, accessing data in a transparent and efficient manner is a major issue, both in operating system design and in microprocessor architecture. In operating systems, efficiently moving pages from disk to memory is crucial; in microprocessor architecture, instruction fetch time plays an important role; on large-scale distributed systems, transferring data files between geographically-separated storage sites, and optimizing data access in supercomputers, have major effects on overall performance. Ken Batcher's half-serious, half-humorous Supercomputer definition interestingly addresses the importance of data placement middleware: "A supercomputer is a device for turning compute-bound problems into I/O-bound problems".

I/O happens to be one of the major bottlenecks for end-to-end application performance especially for peta-scale applications. Data transfer in a distributed environment is prone to frequent failures resulting from back-end system level problems. Besides, we use a dynamic network where data placement middleware needs to adapt to the changing conditions in the environment. Also, heterogeneous resource and different data access and security protocols are some of the challenges the data placement middleware needs to deal with. Although through the use of distributed resources the institutions and organizations gain access to the resources needed for their large-scale applications, complex middleware is required to orchestrate the use of these compute, storage, and network resources between collaborating parties, and to manage the end-to-end processing of data. We focus on data access and data placement problems for Large Scale Collaborative Science. Data-aware computing' is a new design concept integrating every element in the system hierarchy based on the data access requirements of a large-scale application [2].

2 Stork: Data Placement Scheduling

In our recent research, we have investigated possible metrics that play important role in making the scheduling decision like priorities of requests, file size, available storage space, network latency, and system load [3]. We have also studied adaptive data scheduling in which global knowledge is used to make more precise and dynamic parameter optimization for faster data transfers [4]. Inspired by prefetching and caching techniques in microprocessors, he have implemented aggregation of data transfer requests in order to increase the throughput especially for transfers of small data files by minimizing the effect of connection and protocol setup time. Besides, data placement scenario in distributed environment is quite different from microprocessors and operation systems in terms of reliability such that data transfers are prune to frequent failures. Failure-awareness and early error detection are some other very important features integrated into the data placement scheduler [5].

Aggregation of Data Placement Jobs: We have successfully applied job aggregation in our data scheduler, Stork [6], such that total throughput is increased by reducing the number of transfer operations. According to the file size and source/destination pairs, data placement jobs are combined and processed as a single transfer job. Information about the aggregated job is stored in the job queue and it is tied to a main job which is actually performing the transfer operation such that it can be queried and reported separately. We have seen vast performance improvement, especially with small data files, simply by combining data placement jobs based on their source or destination addresses.

Reliability and Failure-Awareness: Prior knowledge about the environment, and awareness of the actual reason behind a failure, would enable data placement scheduler to make better and accurate decisions. Network exploration techniques have been studied in order to classify and detect network error as early as possible. We have also successfully developed error detection and classification strategies for failure-aware data scheduling. The data scheduler checks the network connection and availability of data transfer protocol beforehand, with the help of a new network exploration module. These features also enable us to select amongst the available data transfer services provided by a storage site.

Tuning Data Transfers: Latency between interconnects is one of the bottlenecks in data access - not only in distributed systems but also in microprocessor architecture. Similar to memory wall in microprocessors, we see latency wall in data access over high bandwidth connections. We have studied dynamic parameter tuning in wide-area data transfers for efficient utilization of available network capacity and optimized end-to-end application performance. Impacts of parallel TCP streams as well as concurrent data transfer jobs running simultaneously have been explain in [4]. We present an adaptive approach for tuning parallelism level of data placement jobs in distributed environments. The adaptive data scheduling includes dynamically setting parameters of data placement jobs. The proposed methodology operates without depending on any external profiles to adapt to changing network conditions.

Extended Features in Transfer Modules: A real-life data placement scheduler needs to provide several additional features for usability. Recursive copy operation is one of them in which he have implemented data transfer modules for directory and multiple file copy operations. Transfer modules developed in Stork project, are also able to verify the successful completion of the operation by controlling checksum of each file. Moreover, it can recover from a failed operation by restarting failed data transfer operations. The rescue file keeps track of failed and succeeded file transfer operations. In case of a retry from a failure, the scheduler informs the transfer module to recover and restart the transfer using the information from a rescue file created by the checkpoint-enabled transfer module.

3 Use Cases

Today's large-scale scientific applications generate tremendous amount of data. Those data intensive applications usually make use of scratch volumes, which are limited in size, to store output data temporarily in a high performance computational cluster. A general scenario is to use an intermediate storage area and then transfer files to a remote storage for post processing and long term archival. Data Grids provide a distributed environment for indexing and storing large scale scientific data for collaborative science. Staging data in a fast storage space enables fast I/O access. Data generated by a data intensive application is intended to be uploaded to a specific data resource which is usually a remote system [3, 7]. We have developed a distributed data storage system, PetaShare [7–9], that span multiple institutions across Louisiana. Petashare provides a unified namespace using iRODS services and lightweight client tools for efficient and transparent access. Stork [6], data placement scheduler, provides the middleware for orchestrating data placement activities to organize input data retrieval, and uploading of output data.

Scientific applications in Experimental and Computational Fluid Mechanics use Stork for scheduling data placement activities for long term archival. The current dataset contains flow information over 3.6 million grid points distributed in 2088 blocks for a time-span of 100 revolutions of the impeller blade. Therefore, PetaShare is being used as a storage resource of this dataset generated on LONI [11] clusters which can be accessed by several researchers inside and outside of the campus for analysis using new visualization tools. LONI allows fast communication between the visualization servers and PetaShare storage that is essential for efficient sharing and processing of dataset of interest. Stork provides a data migration service to organize the usage of the intermediate storage and also to orchestrate data upload to PetaShare resources.

References

1. M. Balman and T. Kosar, From Micro- to Macro-processing: A Generic Data Management Model, International Conference on Grid Computing, 2007
2. T. Kosar and M. Balman, A New Paradigm: Data-Aware Scheduling in Grid Computing, International Journal of Grid Computing: Theory, Methods and Applications, Elsevier, 2008
3. M. Balman and T. Kosar, Data Scheduling for Large Scale Distributed Applications, International Conference on Enterprise Information Systems, 2007
4. M. Balman and T. Kosar, Dynamic Adaptation of Parallelism Level in Data Transfer Scheduling, International Conference on Complex, Intelligent and Software Intensive Systems, 2009
5. M. Balman and T. Kosar, Early Error Detection and Classification in Data Transfer Scheduling, International Conference on Complex, Intelligent and Software Intensive Systems, 2009
6. Stork: A batch scheduler specialized in data placement and data movement, www.storkproject.org
7. M. Balman and I. Suslu and T. Kosar, Distributed Data Management with PetaShare, The 15th Mardi Gras Conference, 2008
8. PetaShare: A Distributed Data Archival, Analysis and Visualization Cyberinfrastructure for Data-intensive Collaborative Research. www.petashare.org
9. X. Wang, D. Huang, I. Akturk, M. Balman, G. Allen, and T. Kosar, Semantic Enabled Metadata Management in PetaShare, International Journal of Grid and Utility Computing, 2009
10. M. Balman and I. Akturk and T. Kosar, Intermediate Gateway Service to Aggregate and Cache I/O operations into Data Repositories, 7th Usenix Conference on File and Storage Technologies-FAST'09, 2009
11. Louisiana Optical Network Initiative, www.loni.org