RESEARCH ARTICLE OPEN ACCESS

Different Techniques to Transfer Big data: a Survey

Kiran Kumar Reddi[1] Dnvsls Indira[2]
[1]Department of Computer Science, Krishna University, Machilipatnam, Krishna Dt., AP- 521001
[2]Research Scholar, Department of CS, Krishna University, Machilipatnam, Krishna Dt., AP- 521001

ABSTRACT
Now a days the world is moving around the social networks, i.e most of the people are interacting with each other via internet only. Transmitting data via the Internet is a routine and common task for users today. Transferring a gigabyte of data in an entire day was normal, however users are now transmitting multiple gigabytes in a single hour. The Big Data is the combination of structured, semi-structured, unstructured, homogeneous and heterogeneous data. With the influx of big data and massive scientific data sets that are measured in tens of petabytes, a user has the propensity to transfer even larger amounts of data. When transferring data sets of this magnitude on public or shared networks, the performance of all workloads in the system will be impacted. This paper addresses the issues and challenges inherent with transferring big data over networks. A survey of current transfer techniques is provided in this paper.

Key words: Big Data, Grid, Parallel Transfer, NICE

I. INTRODUCTION

Over the past several years there has been a tremendous increase in the amount of data being transferred between Internet users. Escalating usage of streaming multimedia and other Internet based applications has contributed to this surge in data transmissions. Another facet of the increase is due to the expansion of Big Data, which refers to data sets that are an order of magnitude larger than the standard file transmitted via the Internet. Big Data can range in size from hundreds of gigabytes to petabytes. Today everything is being stored digitally. Within the past decade, everything from banking transactions to medical history has migrated to digital storage. This change from physical documents to digital files has necessitated the creation of large data sets and consequently the transfer of large amounts of data. There is no sign that the amount of data being stored or transmitted by users is steady or even decreasing. Every year average Internet users are moving more and more data through their Internet connections. Depending on the bandwidth of these connections and the size of the data sets being transmitted, the duration of transfers could potentially be measured in days or even weeks. There exists a need for an efficient transfer technique that can move large amounts of data quickly and easily without impacting other users or applications. This paper presents different techniques to transfer the data across the globe. This paper concentrated on survey on grids, parallel techniques to transfer data on internet and a new model to transfer big data efficiently across the globe.

II. RELATED WORK

Retrieving large data files (GB, TB, PB) is a complicated and time-consuming process. These long duration transfers could take tens of hours to several days and a normal "one click and wait" method will not suffice. During the course of the transfer, servers may go off-line and network conditions may change that either hinder or stop the transfer completely. The user needs to know how to maintain the data transmission until completion. This paper gives an overview of previously existing techniques to transfer files around the world.

This section concentrates on grid computing to distribute data and parallel transfer techniques.

2.1 GRIDS

Grid computing has emerged as a framework for aggregating geographically distributed, heterogeneous resources that enables secure and unified access to computing, storage and networking resources (1). Grid applications have vast datasets and/or complex computations that require secure resource sharing among geographically distributed systems. The term "Grid" was inspired by the electrical grid system, where a user can plug in an appliance to a universal socket and have instant access to power without knowing exactly where that power was generated or how it came to reach the socket (1). The vision for grids was similar. A user could simply access as much computing power as required through a common interface without concern for who was providing the resources. Currently, grids have not yet reached that level of simplicity. Grids offer coordinated resource sharing and problem solving in dynamic, multi institutional virtual organizations (2). A virtual organization (VO) comprises a set of individuals and/or institutions having access to computers, software, data, and other resources for collaborative problem-solving or other purposes (3).

A grid can also be defined as a system that coordinates resources that are not subject to centralized control, using standard, open, general-purpose protocols and interfaces in order to deliver nontrivial qualities of service (4). Data grids, a specialized extension of grid computing, are responsible for providing the infrastructure and services to access, transfer, and modify massive datasets stored in distributed storage resources (5). They allow users to access computational and storage resources in order to execute data-intensive applications on remote data. The objective of a data grid system is to integrate heterogeneous data files stored in a large number of geographically distributed sites into a single virtual data management system and to provide diverse services to fit the needs of high-performance distributed and data intensive computing (5).

2.2 GRID MIDDLEWARE

The sharing of resources in a grid is facilitated and controlled by a set of services that allow resources to be discovered, accessed, allocated, monitored, and accounted for, regardless of the their physical location (6). Since these services create a layer between physical resources and applications, they are often referred to as Grid Middleware. Every grid has different service requirements, therefore the architecture and grid middleware implementation of every grid can vary. The middleware of many grids is based on the software architecture called the Globus Toolkit (7). The toolkit is a set of libraries and programs that address common problems that occur when building distributed system services and applications (8). It provides a set of infrastructure services that implement interfaces for managing computational, storage, and other resources. The Globus Toolkit provides all of these services and it is left to grid administrators to determine whether or not to include certain services in their grid implementation. These are a few of the well known and widely used grids that deploy the Globus Toolkit: TeraGrid, Open Science Grid, EGEE, Worldwide LHC Computing Grid (WLCG) (8), China National Grid, UK National Grid Service and NAREGI (9). The architecture of the Globus Toolkit contains several components, each of which is responsible for different grid functions. A few of these services are (7):

- Grid Resource Allocation and management (GRAM) - This service initiates, monitors, and manages the execution of computations on remote computers. It allows a user to specify: the quantity and type of resources needed, the data sets required for their computation, the executable application to be run, the necessary security credentials, and the job persistence requirements.
- Data access and movement - The reliable file transfer (RFT) service is provided to ensure that data is successfully transferred from one location to another.
- Replica Management - This service keeps track of all replicas and their content using a replica location service (RLS) and a data replication service (DRS).
- Monitoring and Discovery - Multiple services collect and process information about the configuration and state of all resources to enable monitoring of system status.
- Security - Services establish the identity of users or services (authentication), protect communications, and determine who is allowed to perform what actions (authorization), as well as manage user credentials and maintain group membership information.

There are several grid applications available for moving data from one location in a grid to another. The most widely-used data movement tool, which is also a component of the Globus Toolkit, is called GridFTP (10; 11). It is an extension of the File Transfer Protocol (FTP) and was designed specifically for grid environments. Several data retrieval techniques developed specifically for retrieving large files in grid computing environments. The sizes of data files requested in grids are much larger than normal web data requests. It is not uncommon for a grid data file size to be measured in gigabytes or terabytes. Users want to be able to download these files as quickly as possible, by any means necessary. Since utilizing a single server can be limiting, retrieving data from multiple servers in a parallel (also known as data co-allocation) has been suggested as an alternative.

III. PARALLEL TRANSMISSION TECHNIQUES USED ON THE INTERNET

Rodriguez and Biersack present mechanisms for parallel access to data on the Internet (11). They develop two different parallel-access schemes: history-based and dynamic. The goal for all of their schemes is to balance the load amongst all available servers by allocating a workload to each replica that is proportional to its service rate. The authors state that parallel access has additional overhead in comparison to a single access. The additional overhead occurs when multiple connections are opened and extra traffic is generated to perform block requests. In order to minimize these overhead costs, these techniques should only be utilized for larger files. Their history-based technique utilizes a database with information about previous rates from the different servers to the receiver in order to estimate future transfers. Using these estimates the algorithm assigns varying portions of the file to each replica with the goal that all servers will finish transferring the portions at the same time. The authors evaluate their history-based technique using live webservers on the Internet distributed across the world. Due to the presence of other Internet traffic, the authors found that the performance of their technique varied at different times of the day. They found that network conditions rapidly change and estimating the transfer rate to every server using past histories results in poor estimates. Their results show that during peak traffic times when transfer rates vary dramatically and historical information is not a good indicator of future performance, their history-based parallel access technique has higher download times than clients accessing a single server. In response to the performance of their history-based technique, the authors develop a dynamic technique that adjusts to changing network conditions. Their dynamic technique divides the desired file into a fixed number of equal sized blocks. The client requests one block from every replica. When a server completes a request, another block is assigned. When there are a small number of blocks outstanding, idle servers are requested to deliver blocks that have already been assigned to another server, but have yet to be received. There will then be multiple servers working on the same requests. The authors state that the bandwidth wasted on overlapping these requests is smaller than the worst-case scenario of waiting for the slowest server to deliver the last block. To further enhance the performance of their technique, they utilize TCP-persistent connections between the client and every server to minimize the overhead of opening multiple TCP connections. They also propose pipelining the requests to each server in order to decrease inter block idle times. With pipelining, a new block request is sent to a server before the previous block request is completely received. In the evaluations of their dynamic technique, the authors find that there is a significant speedup in comparison to a single server access. Since the dynamic technique is not relying on historical information and can adapt to changing network conditions, it has greater performance than requesting data from a single server even under peak traffic conditions. They also observe that the transfer time of a dynamic parallel access is very close to the optimum transfer time. Utilizing request pipelining, the authors demonstrate that their technique would be almost equal to the optimal transfer time. These advanced retrieval techniques allow users to utilize multiple resources simultaneously. These advanced techniques provide improved performance for users, however they are quite complicated to implement and use. They require significant user involvement and require multiple user decisions that can dramatically affect the performance of the transfer. A user needs know how in order to make these techniques function properly and efficiently. Another configuration option that is frequently left for the user to determine is segment size. In some advanced techniques, the data file is divided into small portions called segments. The segment size is often left for the user to decide and the size chosen can affect the performance of the transfer. Determining the appropriate segment size is not a simple task. If the size is too small, a server may receive hundreds to thousands of requests for portions of a single file. This will result in longer disk service times at a server, as the number of users increases. A server's storage system can best service requests if it has greater knowledge of a user's workload. It can better schedule reading from the hard disks, as well as take advantage of pre-fetching and caching strategies.

IV. NICE MODEL

A new, NICE Model for Big Data transfers, which is based on a store-and-forward model instead of an end-to-end approach, is presented. This nice model ensures that Big Data transfers only occur during low demand periods when there is idle bandwidth that can be repurposed for these large transfers. Under this model, Big Data are transmitted when the Internet traffic at the senders LAN is low. If the Internet traffic at the receivers LAN is high at this time, then the data are stored at a staging server and later transmitted to the receiver. Similar to the nice command in Linux, a transfer tool based on the nice algorithm, gives itself low priority and is nice to other applications using the Internet. The overall goal is to develop an application that can transmit big data via the Internet for all users with in any campus.

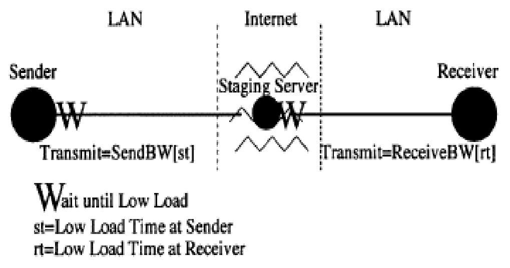

Fig 1: Nice Model architecture

In order to avail of maximum installed bandwidth without impacting other users, the key is to open multiple transmission streams during low traffic. If the sender and receiver are in the same time zone then a direct transmission from sender to receiver is feasible. If the sender and receiver are in different time zones, then the low traffic periods at the two end points do not coincide. In this instance, the file is transmitted from the sender to one or more staging server(s), placed in the Internet zone. Depending on the Internet configuration between the sender and receiver, the file may be transmitted to a single staging server via multiple streams or the file may be divided and parts of the file are transmitted concurrently to multiple staging servers. When the receiver's LAN traffic is low, the file can be transmitted from the staging server(s) to the receiver. A file transmission tool such as GridFTP could be used for the transmission from the sender to the staging server(s) and from the staging server(s) to the receiver.

V. CONCLUSION & FUTURE WORK

Big data transfers via the Internet are not a commonplace task for most users today. Currently, there are no tools to facilitate these kinds of transmissions. The task of transferring massive amounts of data across the country or even the globe is a challenging and daunting undertaking for any user. As the popularity of distributed storage propagates and the amount of scientific data continues to surge, the demand for big data transfers will grow at a tremendous rate. The existing tools for moving large amounts of data are based on the parallel model, which is designed to grab as much bandwidth as possible by opening concurrent data streams. This greedy approach may be good for a single user's transfer, however it is not scalable for multiple users on a shared network. The entire system suffers when users attempt to grab bandwidth. Parallel system supports multiple servers to transfer data via internet but implementation of that system is very difficult. The Nice model can be used for big data transfers. Under this model, these transfers are relegated to low demand periods when there is ample, idle bandwidth available. This bandwidth can then be repurposed for big data transmissions without impacting other users in the system. Since the nice model uses a store-and-forward approach by utilizing staging servers, the model is able to accommodate differences in time zones and variations in bandwidth. There is a lot of challenging issues like network protocols, security, compression techniques, routing algorithms, Traffic Monitoring and bandwidth management devices. New algorithms are required to transfer big data around the globe by concentrating all the above issues.

REFERENCES

[1] FOSTER, I., AND KESSELMAN, C. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[2] FOSTER, I., KESSELMAN, C., AND TUECKE, S. The anatomy of the Grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications 15,* 3 (Fall 2001), 200-222.

[3] LAURE, E., FISHER, S. M., FROHNER, A., GRANDI, C., KUNSZT, P. Z., KRENEK, A., MULMO, O., PACINI, F., PRELZ, F., WHITE, J., BARROSO, M., BUNCIC, P., HEMMER, F., DI MEGLIO, A., AND EDLUND, A. Programming the grid with glite. *Computational Methods in Science and Technology 12*, 1 (2006), 33-45.

[4] FOSTER, I. What is the grid? a three point checklist. *GRIDToday* (July 2002).

[5] XLAO QIN, H. J. *Data Grids: Supporting Data-Intensive Applications in Wide Area Networks.* 2006, pp. 481-494.

[6] LAURE, E., HEMMER, F., PRELZ, F., BECO, S., FISHER, S., LIVNY, M., GUY, L., BARROSO, M., BUNCIC, P., KUNSZT, P. Z., DI MEGLIO, A., AIMAR, A., EDLUND, A., GROEP, D., PACINI, F., SGARAVATTO, M., AND MULMO, O. Middleware for the next generation grid infrastructure. *Computing in High Energy Physics and Nuclear Physics* (October 2004), 826.

[7] FOSTER, I. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing* (2006), pp. 2-13.

[8] Worldwide lhc computing grid (wlcg). *http://lcg.web.cern.ch/LCG/.*

[9] Naregi: National research grid initiative, http://www.naregi.org, 2009.

[10] ALLCOCK, W., BESTEE, J., BEESNAHAN, J., CHEEVENAK, A. L., FOSTEE, I., KESSELMAN, C., MEDEE, S., NEFEDOVA, V., AND STEVEN, D. Q. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *IEEE Mass Storage Conference* (2001).

[11] ALLCOCK, W., BESTEE, J., BEESNAHAN, J., CHEEVENAK, A. L., FOSTEE, I., KESSELMAN, C., MEDEE, S., NEFEDOVA, V., QUESNEL, D., AND TUECKE, S. Data management and transfer in high performance computational grid environments. *Parallel Computing Journal 28,* 5 (May 2002), 749-771.

[12] RODRIGUEZ, P., AND BIERSACK, E.W. Dynamic parallel access to replicated content in the internet. *IEEE/ACM Trans. Netw. 10,* 4 (2002), 455-465.