Load Distribution in Distributed Parallel Storage System

[1]Harikesh Singh, [2]Shishir Kumar
Jaypee University of Engineering & Technology, Guna, MP, INDIA
harikeshsingh@yahoo.co.in, dr.shishir@yahoo.co.in

Abstract: - A collection of standalone computers is referred to as a distributed program from their users. The required memory expanded at various locations in the form of a Distributed Parallel Storage System (DPSS). The basic feature of the DPSS is that the differences between different computers and the way they communicate are highly hidden from users. As per the rule of DPSS, it should be easy to scale or measure. Shared processing on local networks designed for a single system to run simultaneously on multiple sites. The most widely distributed processing systems contain older software that detects CPUs that do not work on the network and package applications to use them. A DPSS upload system has been developed to support various types of data-intensive applications. The functionality, usability, and possible usability of an indivisible, highly efficient, scalable-distributed data-storage system are described in this paper. A collection of local distributed disk servers works similarly to access large data sets. Used primarily as a network-based system, the structures support collaboration between private resources to provide faster, larger, much-needed storage to support data management, simulation, and computing.

Keywords: Distributed Processing, DPSS, Load distribution, Data-intensive applications, MPPs

1. INTRODUCTION

On distributed computers, hardware and software programs that contain more than one processing or storage object, similar processes, or multiple programs, operating under an unregulated or robust state. Any application used on a distributed computer is divided into simultaneous components for multiple computers connected to the network. A distributed computer is a similar type of computer, but parallel computing is often used to describe a simultaneous process on multiple processors on a single computer. This process requires dividing the system into simultaneous components, but distributed systems must deal with complex situations, multi-level network links, and unexpected network and computer failures [1][2][3].

On the same computer, mathematical calculations are performed simultaneously and large problems can be distributed into smaller solutions. There are various types of compatible computers: low level, learning level, data, and job similarity. Similarities have been observed for many years, especially in high computer performance, but interest in it has only recently grown due to physical barriers that prevent wave measurement. If we focus on the same computer architecture that was occasionally used alongside traditional processors, to speed up certain tasks. The same computer is like a lot of things to process at the same time to solve a problem. All we can do is keep it simple or increase the frequency of the clock which lowers the time and takes to do the command. As frequency increases, operating time decreases for all systems included in the calculation. A processor that integrates multiple processing units known as a multicore processor [4][5][6][7].

Equal Shared Storage System contains things that users are working on (either individuals or programs) who think they are dealing with a single system. This means that the private sector needs to work together or harmonize accurately should it happen. The feature is a direct result of having independent computers, but at the same time, we hide how these computers participate in the whole system. The Distributed Parallel Storage System will generally be available on a continuous basis although perhaps some components may temporarily not work on current distributed processing practices.

2. RELATED WORK

Computational data grid always requires a large amount of data in collaboration with the computer services to be run and ensure that data is always available whenever computer census begins on the Internet today is a major challenge[8][9]. Lawrence Berkeley National Laboratory (LBNL) has developed a highly efficient data storage system called the Distributed Parallel Storage System (DPSS) [10][11]. The concept behind the server queue is multiple disks for multiple hosts which work in parallel and generate high-speed data streaming for multiple clients. DPSS is specially designed to access large data objects by remote customers over a wide area network. DPSS has strategies for achieving maximum data transfer over a WAN. LBNL and ANL have worked together to roll out these channels to DPSS using them in the construction

of a high-density external network server using an improved version of FTP called GridFTP [12].

3. SYSTEM ARCHITECTURE

To achieve maximum performance many levels of similarity are created at the level of disks, controllers, processors / memory banks, servers, and network. DPSS is an efficient "block" server that can provide information to applications installed anywhere on the network installed. A medium-speed disk, many low-cost servers use the network to integrate their data stream. The data blocks are dispersed in such a way that multiple system objects run simultaneously to satisfy a given application on both disks and servers. This strategy allows a large collection of disks to search for the same, with all servers sending the leading data to the application in the same way. DPSS at the application level is a persistent storage of data objects, and at the storage level it is a logical server. Typically, data migration involves the transfer of "third-party" from storage servers to direct data consumption. Physical address addresses are translated by these applications, and servers deliver information directly to the application. Used primarily as a network-based repository, the structures support interoperability between independent resources to provide fast, large, much-needed storage space to support data handling, simulation, and computing in a wide range of ATMs in an ATM network. . The following are models based on a distributed system:

A. Client-Server Model

In client-server model, many servers are participating for storing and accessing the information and through any number of clients, we can access the information whenever it is required. Effectively, the server provides uniforms access and published an interface that provides services, and the client uses these services as needed. A widely used client-server application is FTP, a File Transfer Protocol, used to transfer files between computer devices.

B. Peer-to-Peer Model

Each client and server performs simultaneously and distributed all the items equally for sharing the information in peer-to-peer model. The data elements stored in these servers are accessible randomly with more availability and shared the information. In this model, no server is a central instead it provides a specific structure that allow some specific components which are responsible for handling data from other resources.

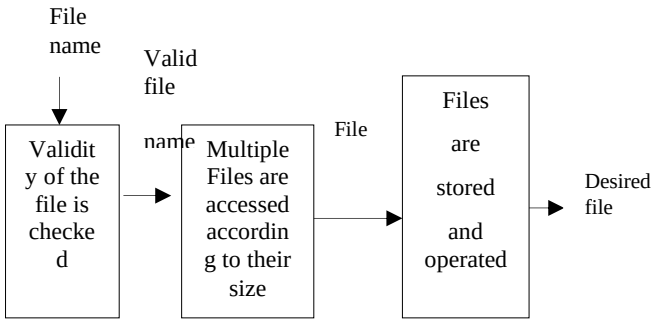

Fig 3.1 Basic functionality of system

Other examples of peer-to-peer programs are USENET, Bit Torrent-based applications, and file-sharing grids used for bioinformatics programs. With the help of Figure 3.1 we show that the client requested file is searched by the server computers. If the requested file is present, it is returned to Fig. 3.2 The central system is connected via a multi-compatible system so that the user is provided with multiple input of various loads, automatically distributing the load evenly as a last reserve in the linked system.

4. PERFORMANCE EVALUATION

The system is implemented with a client and server model where several files differ in their client size and the data is stored on the same computers on a central DPSS server. DPSS distributes these files according to the availability of storage space in the corresponding storage system and if the data is successfully stored then good acceptance is allowed for negative reception to the client. If there is a split or power failure that occurs then the data is also saved.

Server and client are not hardware components that can be programs running on the same computers or on different computers here, different computers start to work as distribution occurs while the same processing is used to decrypt the entire system. A DPSS server containing 2-5 standard SCSI disk drives and a fast network interface capable of 10 to 20 Mbyte / s. So using several DPSS servers one can build a data pipeline very quickly to satisfy almost any application. For example, four server configurations should be able to deliver about 50 Mbyte / s. To perform this function and operation of the algorithm system as described below:

Step 1: Create a socket object
Socket nodesocket = new socket (“127.0.0.1”, 5002);
Socket clientsocket = new socket (“127.0.0.1”, 5000);

Step 2: Declare two objects one each of printscreen and bufferedreader classes
Printscreen out = null and bufferedreader in = null
Step 3: Associate the printstream and buffered reader objects to socket:
out = new printstream (clientsocket.outputstream ());
in = new bufferedreader (new inputstreamreader (clientsocket.getinputstream()) ;
Step 4: Declare another object of the bufferedreader class to associate with the standard input so that the data entered at the client side can be sent to the server:
Bufferedreader stdin = new bufferedreader (new inputstreamreader (system. in)); String str;
While ((str = stdin.readline ( )).length ()! =0) {System.out.println (str) ;}
Step 5: Close the output stream:
Step 6: Close the input stream;
Step 7: Close the standard input output stream:
Step 8: Create a server socket using the constructor of serversocket class
Serversocket = new serversocket (5000)
Step 9: Now the listening of the client request is established by run and accept method.
Step 10: Server is started with the main ( ) method.
Step 11: Reading input from and Sending output to the Client.
Step 12: Now the close the connection related to the server.
Step 13: Creation of the thread takes place with the database.
Step 14: The operating system assigns resources to the thread when the start () method is called, hence confecting Runnable state.
Step 15: If one of the clients is in sleep () state or wait state then that stage will be referred to Not Runnable.
Step 16: A thread method terminates when the run () method exits or stop () method is invoked.
Step 17: Input files to the server are granted by the client using the receive () method.
Step 18: Selection of files takes lace according to the FCFS algorithm.
Step 19: Splitting of the files takes place according to any variable sizes.
Step 20: Now the distribution had took place accordingly when different parts of file are transferred to the nodes.
Step 21: Parts of the file will recombine as in form of downloading and uploading has been done.

The result of this system model specifies that spitted files are stored on the different system and then they are retrieved to form a single file, i.e. the files are checked correctly on the different systems and if errors are found, they are reported correctly. The intermediate result or step by step result is shown accordingly:

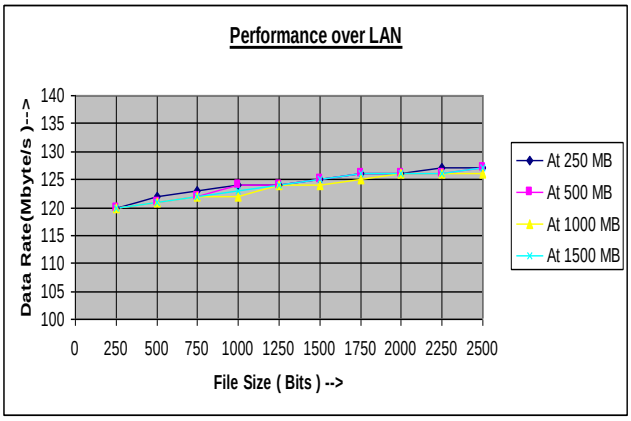

Fig 4.1 Performance over LAN Files to be split according to their size.

- Data is stored on the parallel computers.
- A positive acknowledgement is received (if data stored successfully).
- A negative acknowledgement is received (if data stored unsuccessfully).
- Replica of data is stored as a backup.

As described through above algorithm has been implemented this system for the performance evaluation over LAN having multiple files of different sizes like 250MB, 500MB, 750MB, …2500MB etc. at client system which is further fragmented according to different segments like 250MB, 500MB, 1000MB, 1500MB and results the date transfer rate in 120MByte/s, 122MByte/s, ….127MByte/s for LAN, and it gives the best performance at 126MByte/s as shown in fig 4.1.

As per the architectural model, the networking node in this system can be further enhanced, so we have implemented the above algorithm for WAN having same fragment size over the multiple file of different sizes like 250MB, 500MB, 750MB, …2500MB etc. and for WAN the best performance achieve at 72Mbyte/s as shown in fig 4.2.

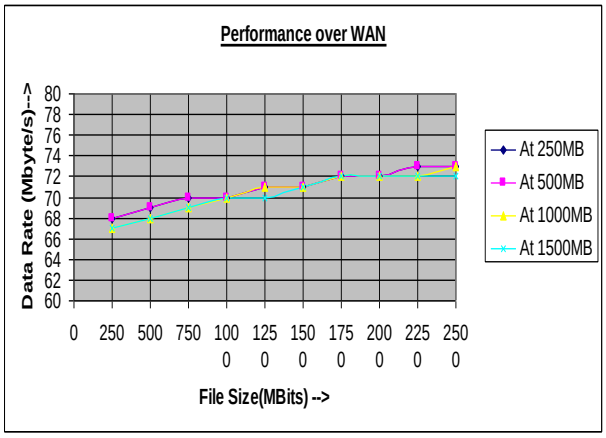

Fig 4.1 Performance over WAN

CONCLUSION AND FUTURE WORK

A compatible shared storage system includes various names such as deadlocks, redundancy, network protocol, etc. The old model works by relying on elements that can be defined by load sharing i.e. when one of the objects is affected, the whole system is blocked. To overcome this issue, the distribution of loads is taken care of to increase the independence between the elements. Other technical difficulties such as job losses, exclusion, are also major barriers to retention. To overcome this problem we will face challenges to get the best result. One of the main purposes of building a DPSS system is to get distributed so that if one of the parallel systems fails, again, the whole system will work. The main use of this system is to detect diversity and growth. In addition, on a security basis, the use of this function will work better. This program provides output based on a distributed distributed storage system.

Eventually the program will split the files and check for errors. After that it will merge into one file. However, some files need to be edited when they include a paid file that does not match the original file. An example of this file could be any JPG file. In addition it can be made more user friendly by adding various options that can help the user. An example related to this situation includes the option to delete that can be added. Many more features can be added over time. Application speed can be increased using a different application algorithm. In addition security can be improved with the use of cryptography. It is used for limited means; some different resources can be added. The DPSS project is fully operational and can be downloaded from http://www-didc.lbl.gov/DPSS. While minor adjustments are still ongoing in DPSS, the most worrying aspect of the job is looking at how high bandwidth data transfer requirements can be integrated during load distribution over the same storage system. Attempts to integrate into the areas of file duplication; stage, temporary storage, location visibility etc. still under investigation. In addition, the use of robust data from performance monitoring is considered as a response function to live transfers. It can strongly conclude that monitoring data can contribute to several levels, such as which link to use, which storage resource you can use in the duplicate data set, and whether data transfer to a computer or vice versa.

REFERENCES

1. M. Blaum, J. Brady, J. Bruck, & J. Menon, “EVENODD: Effective Disk Tolerance Scheme for RAID Architectures”, IEEE Transaction on Computers, vol.44, no.2, pp.192-202, Feb 1995.
2. B. Tierney, W. Johnston, J. Lee, & G. Hoo, “Performance Analysis in High-Speed Wide Area IP at ATM Networks: Top-to-Bottom End-to-End Monitoring”, IEEE Network, vol.10, issue3, July 1996.
3. W. E. Johnston, B. Tierney, J. Feuquay and T. Butzer, “Distributed Parallel Storage Architecture and Its Potential Use at EOSDIS”, NASA Mass Storage Conference, Goddard Space Flight Center, April, 1995.
4. Y. Deswarte, L. Blain, & JC Fabre, Intrusion Tolerance in Distributed Computing Systems, Proc. IEEE Symp. Security and Privacy, pages 110-121, Oakland, Calif., 1991.
5. S. Ghanderharizadeh & L. Ramos, “Continuous Restoration of Multimedia Data Using Parallels”, IEEE Transaction Details and Data Engineering., Vol. 5, no. 4, pp. 658-669, Aug 1993.
6. S. Ghanderharizadeh, L. Ramos, Z. Asad, & W. Qureshi, “Object Placement in Parallel Hypermedia Systems”, in Proceeding of 17 International Conference on Big Data Foundations, pp. 243-254, Sep 1991.
7. D. Agarwal, W. E. Johnston, S. Loken, & B. Tierney, “Tools for Building Virtual Laboratories”, Lawrence Berkeley Lab, 1995.
8. V. Jacobson, R. Braden, & D. Borman, “TCP Extensions for High Performance, Internet Engineering Task Force”, RFC 1323, May 1992. Retrieved from URL: http://ds.internic.net/ds/ dspg1intdoc.html).
9. B. Lamparter, O. Bohrer, W. Effelsberg, & V. Turau, “Multimedia Data Transfer Errors Error”, Technical Report-93-009, International Conference on Computer Science, Berkeley Institute, California, Dec 1993.
10. S. Lau & Y. Leclerc, “TerraVision: A Terrain Visualization System”, Technical Note 540, International Conference on SRI, Menlo Park, California, Mar. 1994.
11. A. J. McAuley, “Broadband Reliable Communication Using Burst Erasure Correction Code, in Proceeding of SIGCOMM, Philadelphia, 1990.

12. D. Patterson, R. Gibson, & R. Katz, “RAID Case: Unnecessary Arrangement of Inexpensive Disks”, in Proceeding of ACM SIGMOD Conference, pp. 109-116, Chicago, May 1988.

13. D. Powel, “Delta-4: A Generic Architecture for Dependable Distributed Computing”, Springer-Verlag, 1991.

14. M. Izard, M. Budiu, Y. A. Birrell, Distributed Parallel Storage System: A fast-paced ATM WAN to support multiple types of data enhancement applications, March 2007

15. X. Shen, A. Choudhary, & C. Matarazzo, “A Distributed Multi-Storage Resource Architecture and I/O Performance Prediction for Scientific Computing”, Cluster Computing, vol.6, pp. 189-200, 2003.

16. A. Goscinski, G. Rünger, E. Gabriel, & C. Morin “Topic 8: Distributed Systems and Algorithms, in Lecture Notes in Computer Science, Springer, vol.4128, 2016.

17. Bethel, W., Tierney, B., Lee, J., Gunter, D., Lau, S., “Using High-speed WANs and Network Data Caches to Enable Remote and Distributed Visualization”, in IEEE Supercomputing Procedures Conference, November 2000.

18. F. R. Duro, J. C. Blas, & J. Carretero, “A hierarchical parallel storage system based on distributed memory for large scale systems”, in Proceedings of the 20th European MPI Users’ Group Meeting, pp. 139-140.

19. Yang D., Wu W., Li Z., Yu J., & Li Y., “PPMS: A Peer to Peer Metadata Management Strategy for Distributed File Systems”, Lecture Notes in Computer Science, vol. 8707. Springer, Berlin, Heidelberg, 2014.