International Journal of Computer Applications (0975 – 8887)
Volume 111 – No 5, February 2015

A Survey: Load Balancing for Distributed File System

Shyam C. Deshmukh
Student, ME Computer
SP Pune University
PCCOE, Pune

Sudarshan S Deshmukh
Assistant Professor
SP Pune University
PCCOE, Pune

ABSTRACT

Distributed Systems are useful for computation and storage of large scale data at dispersed location. Distributed File System (DFS) is a subsystem of Distributed System. DFS is a means of sharing of storage space and data. Servers, Storage devices and Clients are on dispersed location in DFS. Fault tolerance and Scalability are two main features of distributed file system. Performance of DFS is measured by response time. Apart from response time there are also other dimensions such as transparency in which performance of DFS is viewed. DFS provides file services with scalability, fault tolerance, availability, minimum response time. The truth behind the minimum response time is good design of load balance algorithm. To improve the minimum response time and utilization of all nodes in DFS cluster it is found static as well as dynamic load balance strategies. In this survey paper Self acting, load balancing for parallel file system, Adaptive loading data migration in distributed file system, Load balancing in distributed multi agent computing systems, Self organizing storage clusters for data intensive applications, User centric data migration in networked storage systems are discussed to study the different load balancing schemes. Adaptive loading data migration is one of the latest solution found in literature survey. Self acting, load balancing (SALB) for parallel file system is for load balancing uses online load prediction methods and is distributed architecture.

General Terms

Self Acting Load Balancing, Adaptive Loading data migration.

Keywords

Load balancing, Distributed file system, SALB, ALDM

1. INTRODUCTION

A Distributed Systems is a collection of loosely coupled machines either mainframe or workstations [1]. Distributed file system is to allow users of physically distributed computers to share data and storage resources by using common file system. DFS is a means of sharing storage space and data. The file system is a subsystem of the operating system whose purpose is to provide long term storage. DFS is a classical implementation of the time sharing model of the file system where multiple users share files and storage resources. Parallel files are also known as a distributed file system. Now a day's concept of DFS is used in the Cloud systems for the sake of storage of data and later processing on it to analyze although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow. Distributed data centers are maintained to serve the data storage. Stored data are used by multiple clients in parallel. Files are the data structure stored on the data server in DFS. These files are more fault tolerant than traditional file system. Replication concept is to provide more availability of data in DFS. Many clients read, create, update files dynamically hence the load on the file increases. A load imbalance situation occurs due to dynamic nature. If load is not balanced, then performance of the system decreases and hence load balance is necessary for distributed file system.

In this paper, a brief survey of load balancing strategies in a distributed file system and also in distributed systems is given. Section 2 is Background study of Design concepts of distributed file system. Section 3 contains different load balancing strategies for distributed parallel file systems and related information. Section 4 contents Comparative discussion of all the techniques described in section 3. And last section 5 concludes the survey on load balancing in DFS

2. LITERATURE SURVEY

Design features of distributed file system are transparency, scalability, heterogeneity, replication, migration. This feature makes DFS different than the traditional local file system. The file system is responsible for controlling access to data on file and for performing low level operations such as buffering frequently used data and issuing disk I/O requests [2][3]. The Goal in designing a distributed file system is to present a certain degree of transparency to the user and the system, such as access, transparency, location transparency, concurrency transparency, failure transparency, replication transparency, migration transparency. Transparency is a means of hiding the details so that user is unaware of the process.

Service is a file specification of what the file system providing to clients. A server is an implementation of file services and run on one or more machines. The file contains name, date and attribute. An immutable file is one that once created cannot be modified. Two kinds of file protection generally used in a distributed file system are capabilities and access control lists. To provide a remote system file service there are two models of operations such as upload / download model and remote access model. In upload/download model read and write operation performed by transferring files to client and from client to server while writing the file. In this model unwanted data are downloaded while reading. This is a drawback. While in case of other model that is a remote access model all operations are performed on the server hence server is accessed for the time file is read or any operation is going on. Drawback of file system is that server load increases.

Sharing of the file system has some semantics and get clear idea about how file system behaves in distributed environment or multiuser kind of access. How to file actually sharing among the multiple users. Sequential semantics and session semantics are two file sharing semantics, It needs to understand. In sequential semantics, If read follows write, the read of that location will return the values just written. If two writes are occurring in succession, the following read will return the results of last write. Sequential semantics can be achieved in distributed system if there is only one server and client do not cache data. In session semantics changes to open file are initially visible only to the process that modified it.

To design a suitable distributed file system, it is important to understand the usage patterns within a file system. Most files are under 10 K bytes in size, this suggests that it may be feasible to transfer entire files, however file system should be supported to large files. Most files have a short life time. It may be a good idea to keep these files locally and see if they will be deleted soon. Files are shared. While sharing is a big issue theoretically in practice it is hardly done. Files can be grouped into different classes, with each class exhibiting different properties. Naming, cache, stateless and state full connection to the clients are design issues in DFS where the designer needs to concentrate.

To improve the performance It should keep in mind above design concepts. That is above things should not be overridden in any case. If file migration is used to balance the load the naming scheme in metadata management should give same transparency to the file system. Load balancing can be achieved with two approaches. First one is Static Load balancing and Second one is Dynamic approach to solve load balancing problem. Static load balancing is performed in the initialization phase of computation[4].Dynamic load balancing is done during the computation process. Dynamic load balancing is further classified according to migration operation as direct and iterative load balancing [5]. In direct load balancing load is a one step process to balance the load while iterative process require more steps to achieve a load balanced state. Iterative dynamic load balancing is the method used for load balancing further classified as Diffusion Method and Dimension exchange method. In Diffusion method surplus lead is diffused from highly loaded nodes to lightly loaded nodes. In this method, the local load balanced state is achieved first and then automatically global state achieved. For hypercube topology, the dimension exchange method is used. This all the above concept is for load balancing in highly parallel distributed computing system. Load balancing in distributed file systems is done with file allocation and file migration strategies[6]. Based on the file migration approach, A dynamic and Adaptive load balancing strategy for parallel file system with large scale I/O servers is latest SALB [7] algorithm dynamic load balancing. Dynamic load balancing is achieved with the two different ways in distributed system such as centralized load balancing and Distributed load balancing. In centralized load balancing method decision for load balancing is taken by a centralized system which controls distributed system. Metadata information, the information about all the nodes in a distributed system is maintained on one of the nodes which responsible to control the architecture. Centralized load balancer uses metadata and decision of file allocation or file migration takes place to balance the cluster.

Load balancing in distributed multi agent computing system publishes one of international journal of publication [8]. In this paper load balancing is achieved with the help of agents. Agents are then migrated through all nodes based on credit value. Credit value factors are decided at initial stage. Value id dynamically calculated. Node selection for migration is decided on the basis of credit value. Adaptive loading data migration is a scheme for distributed file system load balancing [9], in which complete discussion about various kinds of load is documented. Load evaluation in this paper is done very well. Load in this paper is a combination of network load, disk I/O load and Storage load that is utilized disk capacity. For any kind of file Disk Capacity load is remain same but network load and I/O load change according to the current situation of date server access by various clients. User centric data migration in networked storage systems paper proposes solution [10]. In this solution each user or application makes migration decisions based on what is best for its application. When multiple users share storage resources by striping and those devices have performance differences due to network latencies, and disk hardware. User centric migration could improve performance by improving locality. A self organizing storage cluster for parallel data intensive applications is load balancing solution which automatically adapts node addition and deletion from the cluster. The proposed solution in this paper self organizes available storage resources in the following two aspects. First is adaptable to node additions and departure and second is replication healing. This scheme is considered as a variation of chord protocol. The design of this concept exploits the low write-sharing characteristics of the targeted parallel application.

3. LOAD BALANCING IN DISTRIBUTED FILE SYSTEM

In this section, described some selected load balancing techniques for load balancing in distributed parallel file system. Self Acting Load Balancing (SALB) [7] and TH-clufs [11] a special file system for distributed web server providing load balancing solution. SALB algorithm working is described as further. In this paper load is not formulated but Here It is assumed that load is a total storage load, computational load and network load. Storage load can be mathematically formulated as below,

$$Ls = \sum_{i=1}^{n} fs_i \qquad (1)$$

Where

fs is file size stored on server

n- number of files stored in server

Computation load will be number of request processing per unit time. Note that Lc is computational load, in this case.

$$Lc = No. of\ request\ per\ unit\ time \qquad (2)$$

And network load is bandwidth used from available bandwidth denoted mathematically as Ln

Load considered in SALB is

$$L = f(Ls, Lc, Ln) \qquad (3)$$

The load is considered as a function of all three factors because some time author in the literature may be assuming any one factor. Here It is considered that all the load calculation approaches.

SALB is working with four basic steps.

- Online load prediction and load collection
- Distributed load balancing decision
- Optimization for selection of candidates
- File migration

Together all steps above called as SALB approach for load balancing. Following figure shows SALB flowchart.

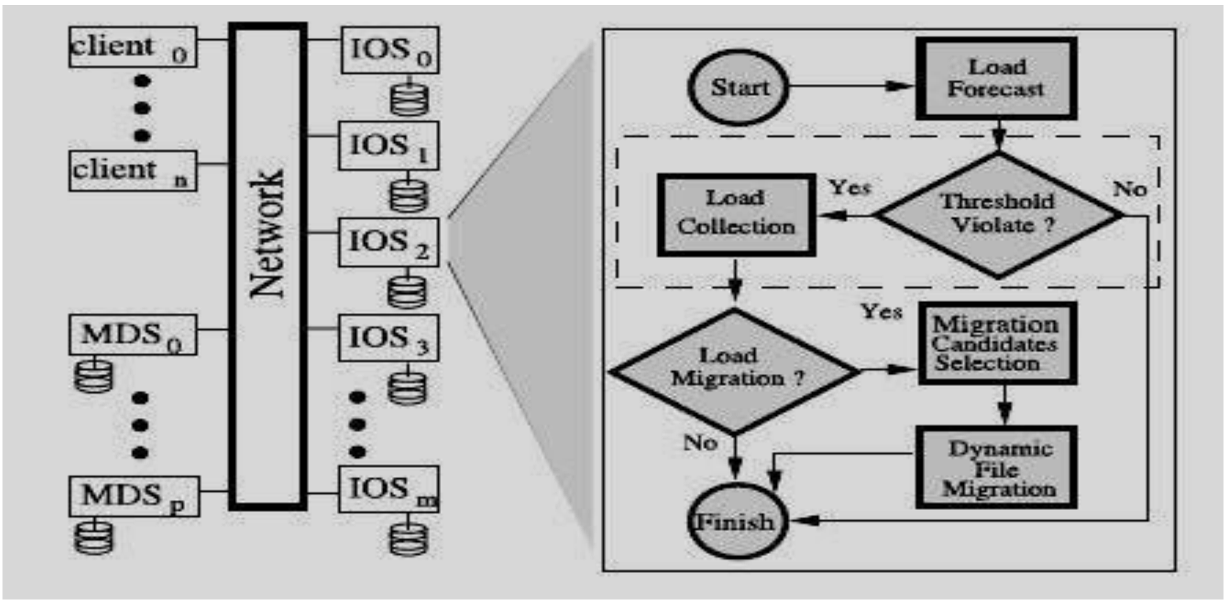

Fig 1: SALB Flow Chart

In online load collection model, forecast load algorithm is used to predict future load from the historical load series model. This prediction model is based on time series model. Each node predicts future load and responds to request of other server for load information. The load information request is sent to collect load from all servers. The distributed load balancing decision means all the nodes in the cluster take part in load balance decision. This approach is the best solution for a scalability problem in distributed systems. In SALB each I/O server should independently make its own decision and also take account the local load and whole system load. Local load calculated with forecast model which predict future load. The whole system load is measured by the efficiency of load balancing. (ELB) which is defined as

$$ELB = \frac{\frac{1}{N}\sum_{i=1}^{N}\{La_i\}}{\max_{i=1-N}\{La_i\}} \qquad (4)$$

Where N –Number of I/O servers. Lai is load on ith server.ELB is always falling between zero and one. More closely the ELB approaches to one the more evenly load is distributed. Objective of optimization of candidate selection is to balance the benefit and side effects of dynamic file migration. The last step is dynamic file migration, In this algorithm migration of sub file or file from one node to another node in cluster of I/O sever is achieved. This algorithm (SALB) is evaluated on various analysis tools and more importantly in PVFS architecture which is a parallel virtual file system in Linux environment. Now second paper, In this Design of I/O balancing file system on web server clusters paper, a new approach of load balancing is proposed which solve the problem of locality of reference that is hotspot problem. The system architecture for this research is distributed web server which provides distributed file storage service to clients.

Following figure shows architecture of distributed storage for TH-CluFS.

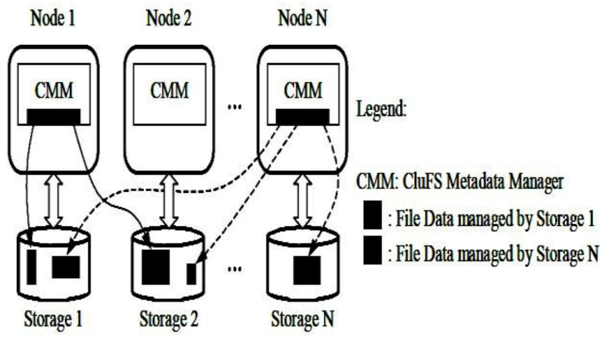
Node 1
Node 2
Node N
CMM
CMM
CMM
...
Storage 1
Storage 2
Storage N
Legend:
CMM: CluFS Metadata Manager
: File Data managed by Storage 1
: File Data managed by Storage N

Fig 2:The storage architecture of TH-CluFs

Above Fig 2 shows TH-CluFS architecture which contains CMM that is the CluFS metadata manager. This architecture is special as it has global namespace and supports real distributed storage granularity. This I/O balancing mechanism solves the hot spot problem by redirecting requests on free servers and migrating files on unloaded servers. Autonomous file migration mechanism and unique cache mechanism are two key mechanisms to solve the load balancing issue presented in this paper.

Adaptive loading data migration is latest solution available in the literature for load balancing in a distributed file system (ALDM) [9], This effective solution because in this paper most of the factors affecting on load balancing are considered. This paper suggests that cannot measure the load of DFS by user requests directly because cache effects on load. So here It is needed an understanding of cache in a distributed file system. Accessing metadata information from the centralized server cannot reflect the system load because system load is not only the capacity of disk used to store a file. In this solution load is measured with network load, disk capacity and Disk I/O load.

The load definition in this reference is given as below

Network load is given as

$$Ni(ni) = \frac{K_N^{ni}-1}{K_N-1} \qquad (5)$$

Where Ni is network utilization of network resources of the data server, which is generated by the average network I/O speed, dividing the network speed of the full capacity in the period.

Disk I/O loading on data server is calculated as given in equation 6

$$Di(di) = \frac{K_D^{di} - 1}{K_D - 1} \quad (6)$$

Where d_i is the disk utilization of data server. It is an average disk utilization in a period of time.

C_i represents the load of disk capacity of data server

$$C_i = \frac{Cu}{Ca} \quad (7)$$

ALDM design as follows

1) The parameters of the system resources on the data server, get the load status on the data server. Supposing the network load of the DSi is Ni, Disk I/O load Di and Disk capacity, load Ci where values of load in the range (0,1) which reflects corresponding resources of data server.

2) After controlling node receiving the monitoring information collected by data server, calculate the server network load, disk I/O load and disk capacity load.Also calculate standard deviation of loads and take them as the load unbalance factor. Average load is calculated and using standard deviation formula load unbalance factor is calculated as

$$\delta = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Ni - \varphi)^2} \quad (8)$$

$$\varphi - Average\ load$$

This is one of load imbalance factors for network load, similarly an imbalance factor for Disk I/O load and Disk capacity load imbalance is calculated.

3) From this three kinds of load imbalance factor load type are decided.

4) The source node is decided on the basis of corresponding load value.

5) All other nodes considered as the destination nodes. Bid price using following formula is calculated to choose an appropriate destination data server to migrate excess load. Files are categorized into three types according to access pattern of the file. That is hot file, warm file, cold file. Bid price for this file are calculated to decide which file to migrate on which node

$$Phi = wsNi + Di + Ci \quad (9)$$

Bid price for hot files. Similarly bid price for warm file calculated using formula

$$Pwi = Ni + Di + Ci \quad (10)$$

And the bid price for cold file calculated only considering disk capacity load.

$$Pci = Ci$$

According to sourthe destination node effect node effect of migration of file is calculated as follows. For hot files

$$Ehi = ws\delta N(Ns - Ni) + \delta D(Ds - Di) + \delta C(Cs - Ci) \quad (12)$$

$$Ewi = \delta N(Ns - Ni) + \delta D(Ds - Di) + \delta C(Cs - Ci) \quad (13)$$

$$Eci = \delta C(Cs - Ci) \quad (14)$$

From this value cost performance ration to decide destination node.

$$\rho hi = Ehi / Phi \quad (15)$$

In this way this ALDM algorithm proposes migration of file to balance the load of the file system. In literature study it is found, another similar work to solve the load balancing issue based on the open queuing network to assignment of non portioned files on the parallel I/O system. Minimum mean response time [12] is achieved with sort partition. Sort partition which assigns to each disk files with similar access time.

4. COMPARATIVE ANALYSIS

In comparative study. The comparison is generally based on centralized or distributed migration decision. Static or dynamic nature of the algorithm, hotspot problem is addressed or not addressed and operation performed to balance the load that is, whether file assigned to a particular node in an optimized way at the time of file creation or migration of files. Apart from this head also distinguish between the schemes based on implementation architecture. Whether replication was considered or not at the time of load balancing migration decision is considered because replication impacts on a distributed file system. If the migration decision is not considered the relation between replication and migration process, then it may lead to unwanted duplication at the same node. And if there is none other chance of destination to migrate then need to migrate extra files so that it's impact on performance of the system. SALB, self acting, load balancing for parallel file system is based of online prediction of load model. Online prediction is a strong point with this scheme. In this scheme every node. Participates in migration activity and hence it is purely distributed in nature. The scheme works at the time of data access and is of dynamic nature, but the load is not only file size that is capacity of disk require to store files. In this scheme author, not clearly mentioned his point of view on load. So It is assumed that the load is not considered as the number of times file is accessed is also kind of loud. It can realize the file is hot accessed by clients and hence load on the particular data server is increased. In SALB file access frequency is not considered that is a hotspot problem is not addressed.

Table 1: Comparative study of different approach

Algorithm	**Centralized/ Distributed**	**File Operation**	**Hotspot Problem**
SALB	Distributed	Assignment and migration	Hotspot problem is not addressed
ALDM	Centralized	Migration	A hotspot problem addressed
TH-CluFs	Centralized	Assignment and migration	The Hotspot problem is addressed

Sort Partition	Centralized	Assignment	The problem is not addressed
User-centric approach	Distributed	Migration	The problem is addressed

All are balancing load on I/O server, which provide file system service to clients. SALB mechanism is distributed architecture while the TH-CluFs and Sort partition are centralized.

TH-CluFs is a distributed file system designed to solve the problem of hot access files. That is a load imbalance situation occurs due to hot access files. In this scheme metadata structure is modified to store details on file. I/O balancing solves the problem of hot access files and redirect the request on free nodes.While all the above strategies are dynamic load balancing. Sort Partition is centralized dynamic scheme based on partition of the load according different loads and this scheme not addressing hot access file problem. A user centric approach in which load migration decision is initiated by users in a distributed manner. This scheme is distributed and problem of hot file access is addressed in this scheme.

5. CONCLUSION

In this paper, literature study is described in detail. It is clear that distributed load balancer helps to improve scalability of the system. The self acting, load balancer is distributed load balancing mechanism but which is based on throughput of each node. This method may be weak in the case of Hotspot problem. Adaptive loading data migration is centralized decision of load migration. Every node sends load to a centralized node, but this method evaluated, load effectively with multiple attributes and as it consider disk I/O load it helps to solve a hotspot problem up to some extent. So Future scope is to design load balancing with distributed load migration decision and solve hotspot problem.

6. REFERENCES

[1] Lin –Wen Lee ,Peter Scheuermann, Radak Vingralek, File Assignment in parallel I/O systems with minimum variance of service time, 1998

[2] E. Levy and A. Silberschatz, Distributed file system : concepts and example, ACM Computing Surveys, Vol. 22, No. 4, December 1990

[3] E. Levy and A. Silberschatz, Distributed file system : concepts and example,

http://www.cs.virginia.edu/zaher/classes/CS656/levy.pdf

[4] Onur Destanoglu, Hydrodynamic based hybrid load balancing IEEE, 978-1-4244 2881, 2008, pp 1-3

[5] Onur Destanoglu, F. Erdogan Sevilgen, Randomized Hydrodynamic Load Balancing Approach, IEEE 1530-2016 ,2008

[6] Bazalel Gavish,Olvia R. Liu Sheng, Dynamic File Migration in Distributed Computing Systems. ACM ,Vol 33 ,Number 2 1990.

[7] Bing Dong Xiuqiao Li ,Qimeng wu,limin xiao,Li ruan, "A dynamic and Adaptive load balancing strategy for parallel file system with large scale I/O servers" ,Journal of Parallel and distributed computing. ELSEVIER 2012.

[8] Maha A Metawei, Salma A. Ghoneim Sahar M Haggag, Salwa M Nassar, Load Balancing in distributed multiagent computing system, Elsevier Aims Shams Engineering journal 2012.

[9] Zhipeng Tan, Wei Zhou, Dan Feng, and Wenhua Zhang , ALDM: Adaptive Loading Data Migration in Distributed File Systems, IEEE transactions on magnetics, vol. 49, no. 6, june 2013

[10] Sukwoo Kang, A. L. Narasimha Reddy, User-Centric Data Migration in Networked Storage Systems, 2005

[11] Wei Liu,Min Wu,XinMing ou, Design of I/O balancing File system on web Server cluster , National 863 High tech program 863-306-ZT01-03-1.

[12] Paul Krzyzanowski,Distributed file system design, Rutgers University 2002