ISSN:0975-9646

Load Balancing in Parallel Network File System using Adaptive Load Data Migration

Ratan Deokar[1] Sagar Hagwane[2] Shreenath Iyer[3] Vivek Labhade[4] Mayur Koli[5]
Department of Computer Engineering
Pimpri Chinchwad College of Engineering
Savitribai Phule Pune University
Pune, India

***Abstract* — pNFS aims to increase the performance of Distributed systems. pNFS, an integral part of NFSv4.1, Promises to bridge the gap between the performance requirements of large, parallel applications and their Interoperability and security requirements. pNFS provides High-performance data access to large-scale storage Systems in both LAN and WAN environments. This paper analyses the problems of load balance caused By data access in parallel file systems and gives an Accurate way to estimate the load of data servers using Mathematical formulae. This paper also analyses ways to Calculate the load on metadata server. The paper gives a Brief information about adaptive loading data migration (ALDM) to balance the load of data servers and parallel NFS (pNFS).**

Keywords*— *Data migration, load balance, load detection, parallel file system.

I. INTRODUCTION

With the rapid growth of distributed file systems, it has become essential to handle the ever increasing issue of the performance of the system with respect to the load. But for typical parallel file systems, such as PVFS and pNFS, in which data is striped to multiple data servers, there exists the cask effect, when some of the data servers load becomes heavy, the clients response time will increase greatly, and furthermore the system performance will decrease. In such cases, it becomes essential to distribute the load evenly among the data servers and make sure that the load of the system is balanced.

With the increase in load on different data servers, the problem of performance degradation, bottleneck and also the availability of resources arises. There are several strategies for improving the performance of system depending on the access of records, prediction etc. This paper considers load balancing and load (data and metadata) migration using a trategy called Adaptive Load Data Migration (ALDM). ALDM strategy focuses on types of files for their distribution such as hot files, cold files and warm files. On that basis there are several formulae which consist of different factors such as disk capacity load, load and network load.

This paper also gives information about parallel Network File System (pNFS) which is an advanced version

Of NFS 4.0 created to help reduce the problem of bottlenecks in the existing file systems. Parallel NFS (pNFS) is a part of the NFS v4.1 standard that allows computer clients to access storage devices directly and in parallel. The pNFS architecture eliminates the scalability and performance issues associated with NFS servers deployed today. This is achieved by the separation of data and metadata, and moving the metadata server out of the data path. By moving the metadata out of the

data path, the data movement is considerably reduced as the clients can directly access the data from the block storage.

A. What is pNFS?

The pNFS (parallel NFS) protocol is being standardized as part of the NFSv4.1 specification to bridge the gap between current NFS protocols (versions 2, 3, and 4) and parallel cluster file system interfaces. Current NFS protocols force clients to access all files on a given file-system volume from a single server node, which can become a bottleneck for scalable performance. As a standardized extension to NFSv4.0, however, pNFS provides clients with scalable end-to-end performance and the flexibility to interoperate with a variety of clustered storage service architectures.

The pNFS protocol enables clients to directly access file data spread over multiple storage servers in parallel. As a result, each client can leverage the full aggregate bandwidth of a clustered storage service at the granularity of an individual file. A standard protocol also improves manageability of storage client software and allows for interoperability across heterogeneous storage nodes. Finally, the pNFS protocol is backward-compatible with the base NFSv4.0 protocol. This allows interoperability between old and new clients and servers.

Using the pNFS protocol, clients gather metadata, called layouts, about how files are distributed across data servers. Layouts are maintained internally by the pNFS server. Once the client understands the file's layout, it is able to directly access the data servers in parallel. Unlike NFSv4.0 whereby a client accesses data via the NFS protocol from a single NFS server, a pNFS client communicates with the data servers using a variety of storage access protocols, including NFSv4.0 and iSCSI/Fibre Channel using the SCSI block command set or the new SCSI object command set. The pNFS specification allows for the addition of new layout distributions and storage access protocols. It also provides significant flexibility in the implementation of the back-end storage system.

The design of pNFS follows three main principles:

1) **Familiar semantics**
   The pNFS protocol must provide consistency and security semantics similar to the base NFSv4 protocol. This simplifies the adoption of pNFS by the diverse set of existing NFS applications.

2) Simplicity
The pNFS extension must be kept simple, yet efficient. The pNFS interface must provide the most basic primitives needed to exploit data parallelism efficiently, while allowing the layering above of additional, more complex semantics.

3) Flexibility
The pNFS protocol must be flexible enough to accommodate a variety of storage architectures. These architectures include a federation of standalone file servers, a mix of a NAS (network-attached storage) file server and RAID storage devices, and a fully transparent parallel clustered file system. Architecturally, the components include one metadata server, some number of data servers, and some number of NFS clients.

B. Operating Environment and Design Characteristics

The metadata server and the data servers of the pNFS used for the purpose of this research are set up on Fedora operating system (Fedora 20) using an Intel i5-4130 processor with 2.4 Ghz Clock speed.
The ALDM algorithm and related load calculation files are developed using Shell script.

C. pNFS Configuration and Setup

Setting up pNFS is a challenging task and should not be underestimated. The main concern with pNFS is that it has not been fully commercialized. Hence, the available versions are just prototypes. During pNFS setup and configuration, care has to be taken that the pNFS utilities are correctly configured.
Below 9 steps are involved in pNFS setup for the data server:

1. Building the code
2. Blkmapd
3. Exporting the filesystem
4. Setting up the BLOCK storage / SAN
5. Export Options
6. Ctl
7. How to Start the server
8. Mount from the client
9. How to verify

Link given in reference section can be used to configure and setup the pNFS server as well as client setup. Configuring pNFS is very hectic job and that is why following are the areas where care should be taken:

- First of all, have a backup of all the packages and codes that you have written, beacause, there is a possibility of losing the data as we need to deal with kernel and operating system.
- While exporting the file system, the disk used on data server must have some signature so gparted tool is very handy. Here care should be taken that the operating system partitions should not be touched and overwritten (in case the system uses two operating systems).
- To map different devices in the network, we need ctl which will map the devices. Here, while building the ctl, a lot of errors occur (if your system isn't compatible). So care should be taken that you have a compiler for compiling c source code and above mentioned operating system(Fedora).

II. LOAD EVALUATION

In distributed file systems, it is not possible to measure the load of the system based on client requests because of client cache. Clients can directly communicate with data server and read and write data after accessing the metadata information for the file rights. So accessing metadata information can not reflect the system load. Accessing data generates the load of data server, and it can also consume resources on data servers, so the resources on corresponding data server determine its service capabilities. Thus, it is efficient to show the load through resource utilization. The utilization of resources reflects the load of data servers, however, it is difficult to compare different computer system resources directly for different characteristics of resources.

For any kind of resource utilization μ_i its Price P_i is

$$P_i(\mu_i) = P_{max}\frac{K_i^{\mu_i} - 1}{K_i - 1}$$

where K_i is the performance degradation caused by resource utilization. The price of a resource is inversely proportional to its utilization. Hence, higher the resource's utilization, lower is its cost.

Load of the system is evaluated by calculating three factors:

1. Network loading of data server
2. Disk I/O loading on the data server
3. Loading of data server adjusted by the capacity

Network loading of the data server is defined as N_i

$$N_i(n_i) = \frac{K_N^{n_i} - 1}{K_N - 1}$$

where n_i the 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 a period.

Simply put, n_i can be calculated as:

$$n_i = \frac{Average\ Network\ IO\ Speed}{Total\ Network\ Speed}$$

Disk I/O loading on the data server is defined as D_i

$$D_i(d_i) = \frac{K_D^{d_i} - 1}{K_D - 1}$$

where d_i the disk utilization of data server, it is the average disk utilization in a period of time.

C_i represents the loading of data server adjusted by the capacity

$$C_i = \frac{C_u}{C_a}$$

where C_u is the capacity used by data server, C_a is the data server total disk capacity.

III. DESIGN OF ALDM

In distributed file systems, load of data server is generated by receiving data requests. But the requests of the clients are complex and changeable, and it is difficult to measure the actual system loads directly through client requests. When data server provides file system services, it needs to consume its own resources. If system load is light, small number of resources is consumed, so the remaining resources can provide more services; if system's load is heavy, more resources are consumed, so there leaves little ability to provide extra services. Therefore the usage situation of data server resources can reveal the load status.

Load migration is done when the system is unbalanced. Load migration occurs from an over loaded server to an under loaded server. The traditional method of load migration is using hot files. Hot files are the most frequently accessed files. However, this method is not completely reliable and thus we migrate load dynamically. For hot-data based data migration, the total number of access bytes of file divided by the file size is usually taken as the cost/performance ratio, which assumes that the cost of data migration is only constructed from the resources consumed during migration, and the effect of migration is determined by the access bytes of the migrated data. Doing migration is to reach load balance; it is evident if we move the same file to different data servers, it may generate different load balance effects.

Different file sizes may lead to different effects when data migration. In distributed file systems, the smallest unit of data is stripe, and then is subfiles. When the migration unit is stripe, it has small overhead and high efficiency, but it needs additional metadata. When the migration unit is sub files, the migration overhead is determined by the size of subfiles, we only need to modify the corresponding data server list stored in metadata. Therefore, we choose sub files as the migration unit in the ALDM algorithm.

Load migration occurs when the value of the network loading of data server or disk I/O loading on the data server or capacity of loading of data server exceeds a threshold value. The threshold values are set before hand and the values that are dynamically computed are compared with these preset values.

IV. ALDM IMPLEMENTATION

We balance system load by data migration, but the traditional data migration is hot data migration. It is not the best choice although it can bring some effect. In ALDM algorithm, we dynamically select migration data to achieve balanced load by using a quantitative method. In the design of ALDM algorithm, the main goal is to balance the network load, disk I/O load and disk capacity load.

The algorithm design is as follows:

1) By monitoring the parameters of the system resources on the data server, we can get the load status of data server. Supposing the network load of DS_i is N_i, disk I/O load is D_i and disk capacity load is C_i; where the values of N_i, D_i , C_i are all in the range of (0, 1), which reflect the corresponding resources usage of the data server. If the value is small, it indicates that the used data server resources are less, if the value is big, it indicates that the used resources are much more.
2) After controlling node receiving the monitoring information collected by data server, we can calculate the data server's network load N_i, disk I/O load D_i, and disk capacity load C_i according to the responding formulas. Besides, we can also calculate the value of δ_N (network load standard deviation), δ_D (disk I/O load standard deviation), δ_C (disk capacity load standard deviation), respectively, and take them as the load unbalance factor. Ψ_N represents the average of data server network load:

$$\Psi_N = \frac{1}{n}\sum_{i=1}^{n} N_i$$

Ψ_D represents the average of data server network load:

$$\Psi_D = \frac{1}{n}\sum_{i=1}^{n} D_i$$

Ψ_C represents the average of data server network load:

$$\Psi_C = \frac{1}{n}\sum_{i=1}^{n} C_i$$

δ_N represents the unbalance factor of data server network load:

$$\delta_N = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(N_i - \psi_N)^2}$$

δ_D represents the unbalance factor of disk I/O load:

$$\delta_D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(D_i - \psi_D)^2}$$

δ_C represents the unbalance factor of disk capacity load:

$$\delta_C = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(C_i - \psi_C)^2}$$

3) According to the type of load imbalance state, we can select the type of data migration technique.
4) After selecting the type of migration, we can select the source data server. The server with the maximum load is selected as the source server.
5) Files are divided into hot files, warm files and cold files due to their accessed frequency.

6) According to the source DS load and destination DS load, we calculate the effect of load balance caused by data migration of per unit data.
7) If the overhead of data migration is feasible with respect to the load of the server, data is migrated from the overloaded server (source DS) to the underloaded server (destination DS).

V. METADATA MIGRATION

In traditional distributed systems, there is only one metadata server. Metadata is nothing but data about data. In particular, metadata server contains the file rights of all the files contained in the block storage along with the pointers to all those files. Metadata server monitors the access to the files requested by the clients in Distributed File System. The metadata server is the backbone of the parallel network file system since it handles the requests of the clients and directs them to the respective block storage. The metadata server facilitates flexibility as the clients can directly access the data from the block storage. In file systems upto NFSv4.0, the data movement was handled by data servers. Hence, data had to be moved from the block storage to the data servers and from the data servers to the clients. However, using a metadata server in NFSv4.1, data movement is halved and block storage access is given directly to the clients. This substantially reduces the data movement and increases efficiency and speed of data access.

However, no matter what kind of load balancing techniques or file system architecture we use, if the backbone of the system is overloaded, the entire sytem is a danger of going down. Metadata sever is the backbone of NFSv4.1 or pNFS. It handles all the requests by the all the clients at the same time. As there is only one metadata server, the question of what happens if the metadata server itself goes down arises. Hence, it is vital to make sure that the metadata server is not overloaded and if it does become overloaded, necessary steps needs to be followed in order to make sure that the entire system does not go down.

A. How Can a Metadata Server crash?

Following are the reasons why metadata server can crash :

1) Metadata can crash when load of handling requests becomes higher than what it can handle without being overloaded. Hence, there arises a need to migrate metadata.
2) If the processor of the metadta server is slow, it will process requests slowly and the number of incoming requests may exceed. Thus, the metadata server queue will be filled with more requests than it can handle.

B. Metadata Migration using ALDM

1) Select appropriate threshold values for network load, disk I/O load and disk capacity which if exceeded determines that the server is overloaded.
2) Determine values of network load (Ni), disk I/O load (Di) and disk capacity load (Ci).
3) Compare the above values with the threshold values.
4) If the values exceed the threshold values, migrate the metadata to an underloaded server using a sender initiated algorithm.
5) Perform steps (2) and (3) after a predetermined time.

VI. CONCLUSION

Hence, the efficiency of a parallel network file system can be improved by adaptive loading migration system.The performance of the system can be increased by using the mentioned load migration techniques. By migrating both the data and the metadata, the chances of system failure is substantially reduced. The system also gives a higher throughput when the adaptive load migration technique is used.

REFERENCES

[1] Zhipeng Tan, Wei Zhou, Dan Feng, and WenhuaZhang, ALDM: Adaptive Loading Data Migration in Distributed File Systems, In Proc. IEEE TRANSACTIONS ON MAGNETICS, VOL. 49, NO. 6, JUNE 2013.

[2] G. Zhang, L. Chiu, and L. Liu, Adaptive data migration in multitier storage based cloud environment, in Proc. 2010 IEEE 3rd Int. Conf. Cloud Computing. (CLOUD), pp. 78155.

[3] K. Dasgupta, S. Ghosal, and R. Jain, QoSMig: Adaptive rate controlled migration of bulk data in storage systems data engineering, in Proc. 21st Int. Conf., 2005, pp. 816827.

[4] S. Kang and A. L. Narasimha Reddy, User-centric data migration in networked storage systems, in Proc. Parallel Distrib. Process. 2008, pp. 112.

[5] J. M. Kunkel and T. Ludwig, Bottleneck detection in parallel file systems with trace-based performance monitoring, in Proc. Parallel Process. Lecture Notes in Comput. Sci., 2008, pp. 212221.

[6] A. Batsakis, R. Burns, and A. Kanevsky, CA-NFS: A congestion aware network file system, in Proc. 7th USENIX Conf. File Storage Technology.

[7] Frank Kargl, Jrn Maier, Stefan Schlott, Michael Weber Protecting Web Servers from Distributed Denial of Service Attacks, in Proc. ACM 1-58113- 348-0/01/0005