Evaluating distributed metadata in HPC

Cristina Rodríguez-Quintana¹, Antonio F. Díaz¹, Julio Ortega¹, Raúl H. Palacios¹ and Andrés Ortiz²

¹ Department of Computer Architecture and Technology, University of Granada, Spain
² Department of Communications Engineering, University of Málaga, Spain
emails: crodriguez@ugr.es, afdiaz@ugr.es, jortega@ugr.es, raulhp@ugr.es, aortiz@ic.uma.es

Abstract

In this paper we present a completely distributed metadata method that provides efficient metadata management and seamlessly adapts to general purpose and scientific computing filesystem workloads. The throughput performance is measured by metadata benchmark and compared with several parallel filesystems. The results show great scalability in metadata benchmark even on a single directory accessed by multiple clients.

Key words: distributed filesystems, HPC, metadata management.

1 Introduction

One of the challenges in high-performance computing (HPC) is process data-intensive applications. The ever-growing amount of data is expected to exceed billions of objects, making metadata scalability critical to overall performance. New efficient storage resources are used when system memory is not enough. Usually, data are often stored in filesystem which are accessed remotely by multiple number of nodes. Some penalties in the accessed data may cause the applications to slow down. For example, NFS, one of the most successful network filesystem, is an effective but not efficient filesystem, since disk writes are often slow. NFS presents a drawback compiling Linux kernel that become a insufferable process. In this way, we can wonder the following: Do the process, which need accesses to data-intensive, can be runned efficiently in network filesystems? The answer is yes, but it depends on important factors in the design of a distributed filesystem.
Metadata stores file and directory information. These can be directory trees, timestamps, attributes and permissions. An optimized metadata management is able to distribute requests among different servers so it can improve its overall performance. Recently, some metadata management approaches have been developed to solve the partitioning problem of the namespace and inodes, but they lack of redundancy at the metadata level.

In this paper, we propose an improved procedure to distribute metadata with redundant capabilities that has been implemented and evaluated on AbFS. This way, all metadata servers can be used as active to achieve an optimal distribution of the metadata management workload.

The development of network filesystem is an actual topic to study since a long time ago. In the last years it has had an improvement of the network systems, CPU speed and reliability of systems, which has led to develop more complex and optimised systems. For instance, the first versions of NFS were stateless due to crashes, system interruptions and network errors what it provided simplified and robust models. Nowadays, networks are more reliable, faster and more efficient and the probability of error is almost null. In this way, redundant systems or RAID have enabled more reliable resources.

Network filesystems, like NFS, are a useful solution for multiple applications but present several problems related to get high performance and scalability. Distributed filesystems are a good alternative because they are able to lead complex storage systems. They present some advantages and some limitations. Distributed filesystems are fundamental high-performance computing for reducing the bottleneck, and providing scalability.

Traditionally, metadata has always been studied in the background. Parallel filesystems have focused on scalable data distribution but as supercomputers increase their computational power is necessary to improve the accesses to large amounts of metadata. There are a growing number of parallel filesystem that follow a distribution metadata approach. However, increasing workload, with not only greater volumes of data but also critically ever-greater number of metadata requests. Metadata operations are very CPU consuming, and a single metadata server is often no longer sufficient, even becoming in a bottleneck of system. Creating filesystems based on multiples metadata servers proving somewhat complexity.

Among the parallel filesystems, Lustre [1] has been one of the most successful. Its architecture has had a single metadata server for a long time. They presented some alternative solution based on active-passive model, but in this model the second server was a copy of primary one and there was not any distribution of workload. Recently they have developed some approach where not only distribute data but also metadata. The latest version includes an early implementation of the distributed namespace (DNE) that allows several MDTs. By default, each directory is created in the same node of its father, but the manager node can move subdirectories among different metadata servers. Furthermore, Lustre supports POSIX semantics.

On the other hand, Ceph [8] is one of the youngest object-based filesystems. It offers
a dynamic management metadata based on current access patterns. Ceph stores both data and metadata on object servers using the same strategy, and these are replicated through on placement groups (PG). Ceph adaptively distributes for managing the filesystem directory hierarchy among the metadata servers using a pseudo-random data distribution random called CRUSH [9], which guarantees a balanced distribution. Initially, a hash function is applied to full pathname of a directory object and the obtained value is used as input to CRUSH. Any component of system is able to calculate the location of any directory object. However, this approach also maintains different issues associated to hashing such as symbolic links, permissions changing and cluster changing.

Moreover, Metadata management in Ceph uses Dynamic Subtree Partitioning [5] which distributes responsibility for managing the filesystem directory hierarchy among metadata servers. Such an approach has a drawback. Since the number of metadata servers can limit the sub-directory number, this method may not be enough to scale well when subdirectories grow excessively [3].

One of more popular filesystems is GPFS developed by IBM. It supports efficient lookup and distributes hugedirs through extensible hashing [3]. Directory entries of a large directory are stored in multiple disk blocks. Given a directory, a hash function is applied to the entry name in order to locate the block containing the directory entry. GPFS guarantees POSIX semantics through a distributed locking mechanism to synchronize access to data and metadata at the same time. However, locking mechanisms limit parallelism and affect to creating operations. Due to GPFS is a block-oriented filesystem may present several limitations related to the level of parallelism.

In [2] [7] [4] several approaches are analysed with more details. Existing techniques go from a coarse-grain approach like Static Subtree Partitioning (used by Lustre [1], Coda [6] etc.) to a fine-grain approach like File Hashing (e.g. table mapping, hash mapping)

Finally, after evaluating different methods, and considering that complex solutions are not usually a good choice, we consider the alternative of an optimized design of our previously proposed AbFS filesystem. AbFS has distributed metadata implementing a hash/table-based model that takes advantage of communication between the servers to achieve better performances. One of goals of AbFS is the scalability of metadata access, and thus the scalability of the entire filesystem.

Experience with previous AbFS versions has helped us to reduce complexity while improving performance and reliability. To achieve this, our system offers an efficient metadata storage approach that combines hash/table and B+ trees, and provides excellent performance thanks to B+ tree speed. In the rest of the paper, Section 2 describes the model. Experimental results are presented in Section 3, and the conclusions in Section 4.
2 Distributed metadata model

Distributed systems may present coherence problems so it is important that they use mechanisms which are able to distribute the information robustly. Distributed filesystems must guarantee the consistency using atomicity in metadata operations. In this way, POSIX semantic is guaranteed. Other important problem is to provide load balance since distributed filesystems must equilibrate their workloads among servers. This distribution is important to provide a balanced used of resources and to get high performance. Moreover, it can determinate problems related to scalability.

AbFS2 metadata uses a client-server model with different elements. Metadata servers manage the metadata information composed by three main components: directory structure (dentries), files and directory information (inodes) and files location (extents). To solve the problems with hash approach, AbFS2 metadata model combines a hash/table and B+ trees. This model presents a redundancy approach to provide reliability based on a group term. Hash/table distributes accesses among multiple groups (G) assigned to metadata servers (MS) by using a hash function based on parent inode numbers and names. The previous version used this table to index the volumes where metadata were stored. The group construct is necessary to simplify the redundant model. Each group stores metadata into two B+ trees, that improve performance due to the O(log n) search complexity of metadata operations. These tables store lookup entries and inode information, respectively. This new approach simplifies data distribution among servers and takes advantage of B+ tree speed. We also prefer B+ trees rather than other binary trees, like RB-trees, because B+ trees group their elements and they offer better performances as disk access is done by using blocks of one or more sectors.

AbFS2 stores name entries and inode into two different tables that provide better hardware link support, inode delete operations when files are deleted and they are still in use, and directory related operations.

AbFS2 uses a delegation table with a fixed size (4096 entries) in order to distribute the namespace, depending on the number of server. It stores the group id, instead of the physical location. This approach reduces the migration penalty.

Information stored in AbFS2s lookup table is indexed by the tuple \{Parent inode number, name\}. Each entry stores the inode number related to the name. Also, this table is used to obtain the entries in a directory to process readdir operations. The second table stores inodes and is indexed by inode number.

This decoupling allows the reduction of penalties due to directory renaming. In HBA [10], when an upper directory changes, it affects to hash values in all files and directories placed under this entry. This occurs because hash is calculated based in the relative full pathname and requires complex solutions such as to store the names of files and subdirectories as part of the metadata of their parent directory, and to update the Bloom filters associated. On the other side, AbFS2 depends only on the inode number of the parent’s
directory and, even when the parent’s directory is renamed, its inode number does not change.

When a client asks a file within in a directory, the full pathname is processed and each name and i-node are solved. This approach presents some advantages: Rename operations are simplified, the lack of needed of storage the full pathname, since each file stored has as a key its name and parent inode number, and simplify the management of access privileges in the tree structure of the directory.

At implementation level, AbFS2 has metadata server in the user space with good performance results. Metadata operations in the user space (e.g. create a file or a directory) are not penalized because each operation does not need an extra kernel call (as other filesystems need) in order to create another entry at lower level (ext3 or ext4). Most storage devices have slow response time, so metadata operations in filesystem have to work asynchronously to achieve good performance.

On the other hand, client is maintained at kernel level to get good performance in metadata operations. It uses different Linux’s Virtual File System (VFS) resources such as inode caches, dentry caches, or buffers to reduce network accesses and server workloads.

3 Experimental results

This section describes the tests performed to analyze the system performance. In that, we evaluated the performance and scalability achieved on a single shared hugedir which are accessed by hundreds of processes to create, get the status and delete files.

Since we have to evaluate metadata operations, is necessary to use a specific and up-to-date metadata benchmark like mdtest. It is used to generate appropriate metadata workloads and include configurations with varying numbers of threads.

Experiments were run on a cluster with 13 nodes. Each compute node has two Intel Xeon E5472 Quad-core CPUs at 3.0 GHz, 16GB of RAM, and GJ0120CAGSP disk. The network card is Infiniband DDR Mellanox MT25418. Each node has installed Scientific Linux 6.3.

The analysis of hugedirs includes different filesystems which were installed on cluster:

- **Lustre 2.5.1**

- **Ceph 0.87**: This is the latest version at this moment. We had several problems when using more than one MDS, like very low performance (about 479 files creates per second, 149504 file stat per second and 1935 file removal per second), MDSs crashes, and memory leaks. When we used a single MDS and a single client, recommended in Ceph documentation, the stat operations were improved while creates and removals operations were slow.
Evaluating distributed metadata in HPC

- beegfs 2014.01.r9

![File creation](image1.png) ![Directory creation](image2.png)

Figure 1: Performance of creation operations with several filesystems

Figure 1 shows the number of files and directories created per second on different filesystems. AbFS offers better performance than other distributed Filesystem and, in fact CEPH is not stable enough to take some measures.

4 Conclusions

In this work, we have presented the design and implementation of a new metadata distribution approach run on AbFS2 and has been compared to other popular network filesystems. All of them use multiple metadata servers. The main key is the way that the approach distribute workload among different servers. The model uses a hash/table that simplifies access and allows dynamic reconfiguration when the number of metadata servers have changed. This approach offers scalability and performance features since resources have not affected when the number of file and directories are increased. This approach, also, guarantees the atomicity of metadata operations by using client and server caches to reduce network traffic and increase operation performance. We evaluated the scalability of system with different number of metadata servers and was compared with popular filesystems like Lustre, Ceph and BeeGFS. Results show that the model, which we present, can improve the performance of most popular network filesystems. Finally, we note that the performance scales with number of metadata servers.
Acknowledgements

This work has been partially supported by European Union FEDER and the Spanish Ministry of Economy and Competitiveness TIN2012-32039 and CAPAP-H5 network TIN2014-53522-REDT. The authors would like to thank FCSCL (Fundación Centro de Supercomputación de Castilla y León) for providing access to a cluster of its supercomputer Calendula.

References


[5] Sage A. Weil Kristal T. Pollack Scott A. Brandt Ethan L. Miller sage@cs.ucsc.edu kristal@cs.ucsc.edu scott@cs.ucsc.edu elm@cs.ucsc.edu. Dynamic metadata management for petabyte-scale file systems.


