International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 5

HBA: Distributed Metadata Management for Large Cluster-Based Storage Systems

M S Nirmala
Lecturer (Senior Grade), Department of Electronics Communication Engineering,
Sakthi Polytechnic College, Erode, Tamil Nadu, India

ABSTRACT
An efficient and distributed scheme for file mapping or file lookup is critical in decentralizing metadata management within a group of metadata servers. This paper presents a novel technique called Hierarchical Bloom Filter Arrays (HBA) to map filenames to the metadata servers holding their metadata. Two levels of probabilistic arrays, namely, the Bloom filter arrays with different levels of accuracies, are used on each metadata server. One array, with lower accuracy and representing the distribution of the entire metadata, trades accuracy for significantly reduced memory overhead, whereas the other array, with higher accuracy, caches partial distribution information and exploits the temporal locality of file access patterns. Both arrays are replicated to all metadata servers to support fast local lookups. We evaluate HBA through extensive trace-driven simulations and implementation in Linux. Simulation results show our HBA design to be highly effective and efficient in improving the performance and scalability of file systems in clusters with 1,000 to 10,000 nodes (or super clusters) and with the amount of data in the peta byte scale or higher. Our implementation indicates that HBA can reduce the metadata operation time of a single-metadata-server architecture by a factor of up to 43.9 when the system is configured with 16 Meta data servers.

Keywords: *HB, Nodes, GPFS, PVFS, Mapping*

I. INTRODUCTION

Rapid advances in general-purpose communication networks have motivated the employment of inexpensive components to build competitive cluster-based storage solutions to meet the increasing demand of scalable computing. In the recent years, the bandwidth of these networks has been increased by two orders of magnitude, which greatly narrows the performance gap between them and the dedicated networks used in commercial storage systems. Since all I/O requests can be classified into two categories, that is, user data requests and metadata requests, the scalability of accessing both data and metadata has to be carefully maintained to avoid any potential performance bottleneck along all data paths. This paper proposes a novel scheme, called Hierarchical Bloom Filter Arrays (HBA), to evenly distribute the tasks of metadata management to a group of MSs. A Bloom filter (BF) is a succinct data structure for probabilistic membership query. A straightforward extension of the BF approach to decentralizing metadata management onto multiple MSs is to use an array of BFs on each MS. The metadata of each file is stored on some MS, called the home MS.

In Login Form module presents site visitors with a form with username and password fields. If the user enters a valid username/password combination they will be granted access to additional resources on website. Which additional resources they will have access to can be configured separately.

In this module we are going to find out the available computers from the network. And we are going to share some of the folder in some computers. We are going to find out the computers those having the shared folder. By this way will get all the information about the file and we will form the Meta data.

In this module we are creating a metadata for all the system files. The module is going to save all file names in a database. In addition to that, it also saves

some information from the text file. This mechanism is applied to avoid the long run process of the existing system.

In this module the user going to enter the text for searching the required file. The searching mechanism is differing from the existing system. Whenever the user gives their searching text, it is going to search from the database. At first, the search is based on the file name. After that, it contains some related file name. Then it collects some of the file text, it makes another search. Finally it produces a search result for corresponding related text for the user.

Here we are using the new approaches called HIERARCHICAL BLOOM FILTER ARRAYS (HBA), efficiently route metadata request within a group of metadata servers. There are two arrays used here. First array is used to reduce memory overhead, because it captures only the destination metadata server information of frequently accessed files to keep high management efficiency. And the second one is used to maintain the destination metadata information of all files. Both the arrays are mainly used for fast local lookup.

A BF is a loss but succinct and efficient data structure to represent a set S, which processes the membership query, "Is x in S?" for any given element x with a time complexity. It was invented by Burton Bloom in 1970 and has been widely used for Web caching, network routing, and prefix matching. The storage requirement of a BF falls several orders of magnitude below the lower bounds of error-free encoding structures. This space efficiency is achieved at the cost of allowing a certain (typically nonzero) probability of false positives or false hits; that is, it may incorrectly return a "yes," although x is actually not in S. A straightforward extension of the BF approach to decentralizing metadata management onto multiple MSs is to use an array of BFs on each MS. The metadata of each file is stored on some MS, called the home MS. In this design, each MS builds a BF that represents all files whose metadata is stored locally and then replicates this filter to all other MSs. Including the replicas of the BFs from the other servers, a MS stores all filters in an array. When a client initiates a metadata request, the client randomly chooses a MS and asks this server to perform the membership query against this array. The BF array is said to have a hit if exactly one filter gives a positive response. A miss is said to have occurred whenever no hit or more than one hit is found in the array.

II. COMPARISON OF DECENTRALIZATION SCHEMES

Many cluster-based storage systems employ centralized metadata management. Experiments in GFS show that a single MS is not a performance bottleneck in a storage cluster with 100 nodes under a read-only Google searching workload. PVFS, which is a RAID-0-style parallel file system, also uses a single MS design to provide a cluster wide shared namespace. As data throughput is the most important objective of PVFS, some expensive but indispensable functions such as the concurrent control between data and metadata are not fully designed and implemented. In CEFT, which is an extension of PVFS to incorporate a RAID-10-style fault tolerance and parallel I/O scheduling, the MS synchronizes concurrent updates, which can limit the overall throughput under the workload of intensive concurrent metadata updates. In Lustre, some low-level metadata management tasks are offloaded from the MS to object storage devices, and ongoing efforts are being made to decentralize metadata management to further improve the scalability.

Some other systems have addressed metadata scalability in their designs. For example, GPFS uses dynamically elected "met anodes" to manage file metadata. The election is coordinated by a centralized token server. Ocean Store, which is designed for LAN-based networked storage systems, scales the data location scheme by using an array of BFs, in which the ith BF is the union of all the BFs for all of the nodes within i hops. The requests are routed to their destinations by following the path with the maximum probability. Panasas Active Scale not only uses object storage devices to offload some metadata management tasks but also scales up the metadata services by using a group of directory blades. Our target systems differ from the three systems above. Although GPFS and Panasas Active Scale need to use their specially designed commercial hardware, our target systems only consist of commodity components. Our system is also different from Ocean Store in that the latter focuses on geographically distributed storage nodes, whereas our design targets cluster-based storage systems, where all nodes are only one hop away.

The following summarizes other research projects in scaling metadata management, including table-based mapping, hash-based mapping, static tree partitioning, and dynamic tree partitioning.

TABLE: 1 Comparison of HBA with Existing Decentralization Schemes

	Hashing based Mapping	Table based Mapping	Static Tree Partition	Dynamic Tree Partition	HBA
Example Systems	Lustre [1], Vesta [30], InterMezzo [31]	xFS [32]	NFS [22], AFS [23], Coda [24], Sprite [33]	OBFS [29]	HBA
Load Balance	Yes	No	No	Need load monitor	Yes
Migration Cost	Large	0	0	Large	0
Lookup Time	O(1)	$O(\log n)$	Slow	$O(\log d)$	O(1)
Memory Overhead	0	$O(n)$	$O(1)$	$O(d)$	$O(n)$
Directory Operations	Slow	Medium	$O(1)$	$O(1)$	Fast

2.1 Table-Based Mapping

Globally replicating mapping tables is one approach to decentralizing metadata management. There is a salient trade-off between the space requirement and the granularity and flexibility of distribution. A fine-grained table allows more flexibility in metadata placement. In an extreme case, if the table records the home MS for each individual file, then the metadata of a file can be placed on any MS. However, the memory space requirement for this approach makes it unattractive for large-scale storage systems. A back of-the-envelope calculation shows that it would take as much as 1.8 Gbytes of memory space to store such a table with 108 entries when 16 bytes are used for a filename and 2 bytes for an MS ID. In addition, searching for an entry in such a huge table consumes a large number of precious CPU cycles. To reduce the memory space overhead, xFS proposes a coarse-grained table that maps a group of files to an MS. To keep a good trade-off, it is suggested that in xFS, the number of entries in a table should be an order of magnitude larger than the total number of MSs.

2.2 Hashing-Based Mapping

Modulus-based hashing is another decentralized scheme. This approach hashes a symbolic pathname of a file to a digital value and assigns its metadata to a server according to the modulus value with respect to the total number of MSs. In practice, the likelihood of serious skew of metadata workload is almost negligible in this scheme, since the number of frequently accessed files is usually much larger than the number of MSs. However, a serious problem arises when an upper directory is renamed or the total number of MSs Changes: the hashing mapping needs to be re implemented, and this requires all affected metadata to be migrated among MSs. Although the size of the metadata of a file is small, a large number of files may be involved. In particular, the metadata of all files has to be relocated if an MS joins or leaves. This could lead to both disk and network traffic surges and cause serious performance degradation.

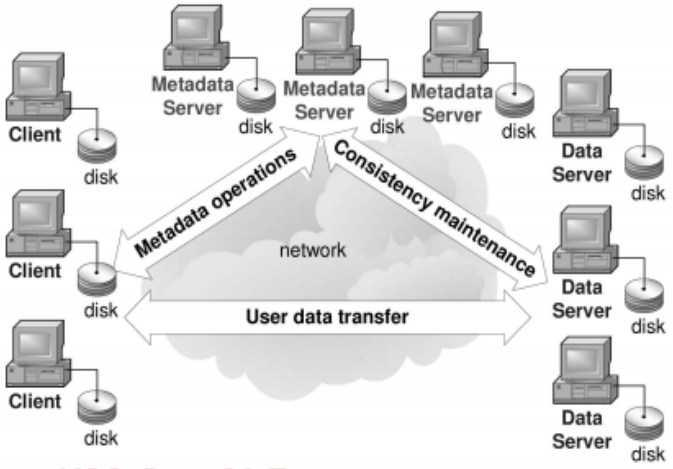

Figure 1: Cluster-based storage architecture.

III. ARCHITECTURAL CONSIDERATIONS AND DESIGN

In this paper, we focus on a generic cluster, where a number of commodity PCs are connected by a high-bandwidth low latency switched network. Each node has its own storage devices. There are no functional differences between all cluster nodes. The role of clients, MSs, and data servers can be carried out by any node. A node may not be dedicated to a specific role. It can act in multiple roles simultaneously. Fig shows the architecture of a generic cluster targeted in this study. In this study, we concentrate on the scalability and flexibility aspects of metadata management. Some other important issues such as consistency maintenance, synchronization of concurrent accesses, file system security and protection enforcement, free-space allocation (or garbage collection), balancing of the space utilizations, management of the striping of file contents, and incorporation of fault tolerance are beyond the scope of this study. Instead, the following objectives are considered in our design:

Single shared namespace. All storage devices are virtualized into a single image, and all clients share the same view of this image. This requirement simplifies the management of user data and allows a job to run on any node in a cluster.

Scalable service. the throughput of a metadata management system should scale with the computational power of a cluster. It should not

become a performance bottleneck under high I/O access workloads. This requires the system to have low management overhead.

Zero metadata migration. Although the size of metadata is small, the number of files in a system can be enormously large. In a metadata management system that requires metadata to migrate to other servers in response to the file system's evolution such as renaming of files or directories, or topology changes involving server arrivals or departures, the computational overhead of checking whether a migration is needed and the network traffic overhead due to metadata migration may be prohibitively high, hence limiting the efficiency and scalability.

Balancing the load of metadata accesses. The management is evenly shared among multiple MSs to best leverage the available throughput of these severs.

Flexibility of storing the metadata of a file on any MS. This flexibility provides the opportunity for fine grained load balance, simplifies the placement of metadata replicas, and facilitates some performance optimizations such as metadata prefacing. In a distributed system, metadata prefacing requires the flexibility of storing a group of sequentially accessed files on the same physical location to save the number of metadata retrievals.

IV. HIERARCHICAL BLOOM FILTER ARRAYS

A BF is a lossy but succinct and efficient data structure to represent a set S, which processes the membership query, "Is x in S?" for any given element x with a time complexity. It was invented by Burton Bloom in 1970 and has been widely used for Web caching, network routing, and prefix matching. The storage requirement of a BF falls several orders of magnitude below the lower bounds of error-free encoding structures. This space efficiency is achieved at the cost of allowing a certain (typically nonzero) probability of false positives or false hits; that is, it may incorrectly return a "yes," although x is actually not in S. A straightforward extension of the BF approach to decentralizing metadata management onto multiple MSs is to use an array of BFs on each MS. The metadata of each file is stored on some MS, called the home MS. In this design, each MS builds a BF that represents all files whose metadata is stored locally and then replicates this filter to all other MSs. Including the replicas of the BFs from the other servers, a MS stores all filters in an array. When a client initiates a metadata request, the client randomly chooses a MS and asks this server to perform the membership query against this array. The BF array is said to have a hit if exactly one filter gives a positive response. A miss is said to have occurred whenever no hit or more than one hit is found in the array.

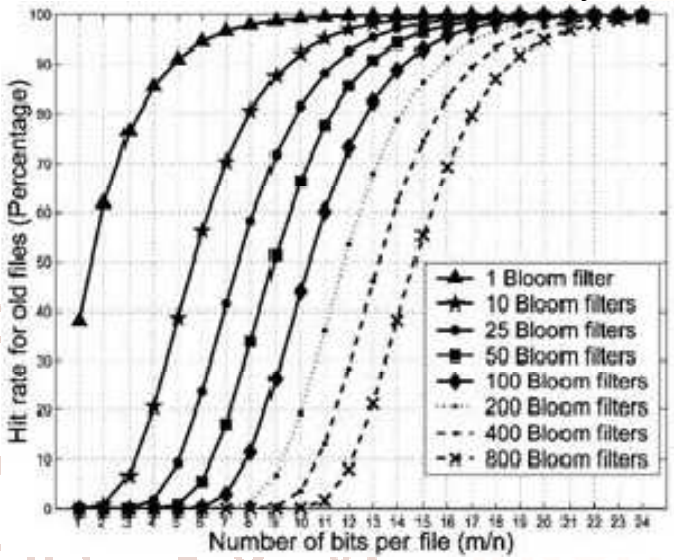

Figure 2: Theoretical hit rates for existing files.

The desired metadata can be found on the MS represented by the hit BF with a very high probability. We denote this simple approach as PBA. PBA allows a flexible metadata placement, has no migration overhead, and balances metadata workloads. PBA does not rely on any property of a file to place its metadata and, thus, allows the system to place any metadata on any server.

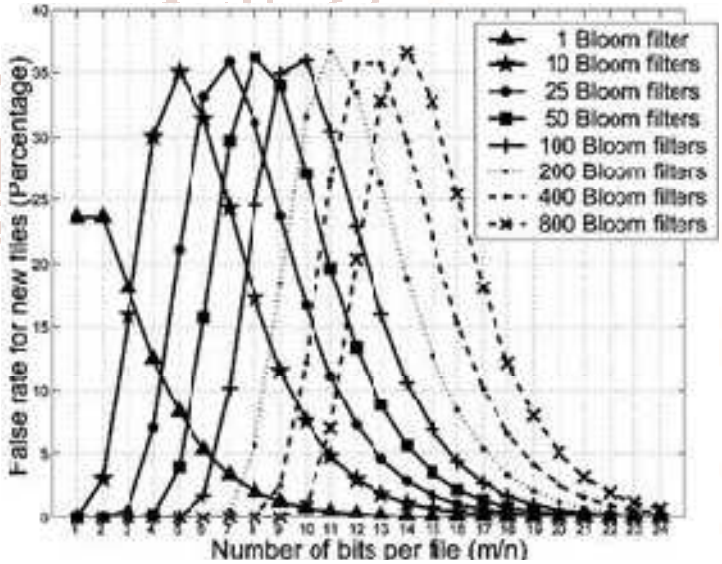

Figure 3: Theoretical false-hit rates for new files.

This makes it feasible to group metadata with strong locality together for prefetching, a technique that has been widely used in conventional file systems. During

the evolution of file system and its cluster topology, not all metadata needs to migrate to new locations. When a file or directory is renamed, only the BFs associated with all the involved files or subdirectories need to be updated. Although a MS leaves or joins the system, a single associated BF is added or deleted from the Bloom arrays on all other MSs. Since each client randomly chooses a MS to look up for the home MS of a file, the query workload is balanced on all Mss. The following theoretical analysis shows that the accuracy of PBA does not scale well when the number of MSs increases.

To achieve a sufficiently high hit rate in the PBA described above, the high memory overhead may make this approach impractical. A large bit-per-file ratio needs to be employed in each BF to achieve a high hit rate when the number of MSs is large. In this section, we present a new design called HBA to optimize the trade-off between memory overhead and high lookup accuracy. The novelty of HBA lies in its judicious exploitation of the fact that in a typical file system, a small portion of files absorb most of the I/O activities. Floyd discovered that 66 percent of all files had not been accessed in over a month in a Unix environment, indicating that the entire I/O accesses were focused on at most 34 percent of the file system. Staelin found that 0.1 percent of the total space used by the file system received 30 percent to 60 percent of the I/O activity. Cate and Gross showed that most files in Unix file systems were inactive, and only 3.6 percent to 13 percent of the file system data was used in a given day, and only 0.2 percent to 3.6 percent of the I/O activity went to the least active 75 percent part of the file system. A recent study on a file system trace collected in December 2000 from a medium-sized file server found that only 2.8 percent and 24.2 percent of files were accessed during a continuous course of 12 hours and 10 days, respectively.

V. PERFORMANCE EVALUATION

We simulate the MSs by using the two traces introduced in Section 5 and measure the performance in terms of hit rates and the memory and network overhead. Since the decentralized schemes of table-based mapping and modulus-based hashing are simple and straightforward and their performance was already discussed qualitatively, the simulation study in this paper will be focused on the schemes of PBA, HBA, and pure LRU BF to obtain quantitative comparison and conclusions.

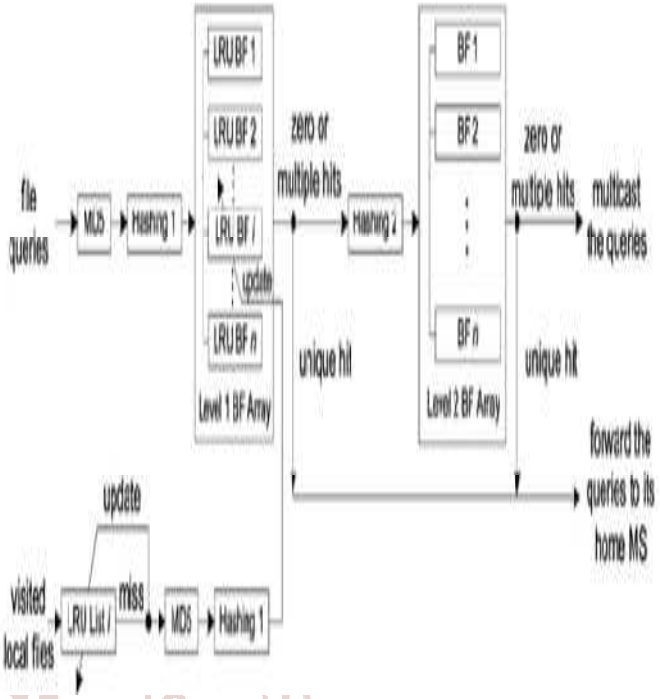

Figure 4: The structure of the HBA design on each MS, which includes two levels of BF arrays.

V. CONCLUSION

This paper has analyzed the efficiency of using the PBA scheme to represent the metadata distribution of all files and accomplish the metadata distribution and management in cluster-based storage systems with thousands of nodes. Both our theoretic analysis and simulation results indicated that this approach cannot scale well with the increase in the number of MSs and has very large memory overhead when the number of files is large. By exploiting the temporal access locality of file access patterns, this paper has proposed a hierarchical scheme, called HBA, that maintains two levels of BF arrays, with the one at the top level succinctly representing the metadata location of most recently visited files on each MS and the one at the lower level maintaining metadata distribution information of all files with lower accuracy in favor of memory efficiency. The top-level array is small in size but has high lookup accuracy. This high accuracy compensates for the relatively low lookup accuracy and large memory requirement in the lower level array. Our extensive trace-driven simulations show that the HBA scheme can achieve an efficacy comparable to that of PBA but at only 50 percent of memory cost and slightly higher network traffic overhead (multicast). On the other hand, HBA incurs much less network traffic overhead (multicast) than the pure LRU BF approach. Moreover, simulation results show that the network traffic overhead

introduced by HBA is minute in modern fast networks. We have implemented our HBA design in Linux and measured its performance in a real cluster. The experimental results show that the performance of HBA is very promising. Under heavy workloads, HBA with 16MSs can reduce the metadata operation time of a single-metadata-server architecture by a factor of up to 43.9

VII. REFERENCES

1. P. J. Braam, "Lustre White Paper," http://www.lustre.org/docs/ whitepaper.pdf, 2005.
2. S. A. Brandt, L. Xue, E. L. Miller, and D. D. E. Long, "Efficient Metadata Management in Large Distributed File Systems," Proc. 20th IEEE Mass Storage Symp./11th NASA Goddard Conf. Mass Storage Systems and Technologies (MSS/MSST '03), pp. 290-298, Apr. 2003.
3. P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, "PVFS: A Parallel File System for Linux Clusters," Proc. Fourth Ann. Linux Showcase and Conf., pp. 317-327, 2000.
4. S. Ghemawat, H. Gobi off, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03), pp. 29-43, 2003.
5. H. Tang and T. Yang, "An Efficient Data Location Protocol for Self-Organizing Storage Clusters," Proc. ACM/IEEE Conf. Super Computing (SC '03), p. 53, Nov. 2003.
6. Y. Zhu and H. Jiang, "CEFT: A Cost-Effective, Fault-Tolerant Parallel Virtual File System," J. Parallel and Distributed Computing, vol. 66, no. 2, pp. 291-306, Feb. 2006.
7. N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su, "Myrinet: A Gigabit-per-Second Local Area Network," IEEE Micro, vol. 15, no. 1, pp. 29-36, 1995.
8. D. H. Carrere, "Linux Local and Wide Area Network Adapter Support," Int'l J. Network Management, vol. 10, no. 2, pp. 103-112, 2000.
9. C. Eddington, "Infinibridge: An Infiniband Channel Adapter with Integrated Switch," IEEE Micro, vol. 22, no. 2, pp. 48-56, 2002.
10. Y. Zhu, H. Jiang, X. Qin, D. Feng, and D. Swanson, "Exploiting Redundancy to Boost Performance in a RAID-10 Style Cluster Based File System," Cluster Computing: The J. Networks, Software Tools and Applications, vol. 9, no. 4, pp. 433-447, Oct. 2006.
11. M. Vilayannur, A. Sivasubramaniam, M. Kandemir, R. Thakur, and R. Ross, "Discretionary Caching for I/O on Clusters," Proc. Third IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGRID '03), pp. 96-103, May 2003