Scalable Distributed File Systems via Correlation-based Metadata Prefetching

First Name
affiliation

Second Name
affiliation

1 Problem and Motivations

Modern distributed file systems, such as GFS[2], HDFS[7], Ceph[8], which consist of three components, metadata servers(MDS), object-based storage devices(OSD) and clients as Fig. 1 shows. Clients perform metadata operations(*open, stat*) directly with MDS and file I/O directly with data server. This architecture decouples metadata I/O from data IO and increases overall scalability. For example, client requests data with pathname /var/log/client.log. By multiple interactions to MDS, client lookups the directory of `root`, `var` and `log` to locate the inode of `client.log` and checks the permission. Once client obtain the inode, it must get the capability to access to data. Then, client will request a lease on this file to MDS. Finnaly, client interact with OSD to operator file I/O. Due to the iterative lookup and checking permission operations, the IOs between clients and MDS are more than OSD. Because overall 50% of IOs are to metadata[6], metadata performance is crucial.

Especially in web content system, loading one web file leads to multiple scripts and images requesting. We analyse the access log[9] of web server and find that there are 23% of requests are correlated. These too frequently correlated metadata fetching increases the traffic to MDS and network. If there is no optimization on metadata fetching, too more traffic will overwhelm MDS and becomes a bottleneck in the system.

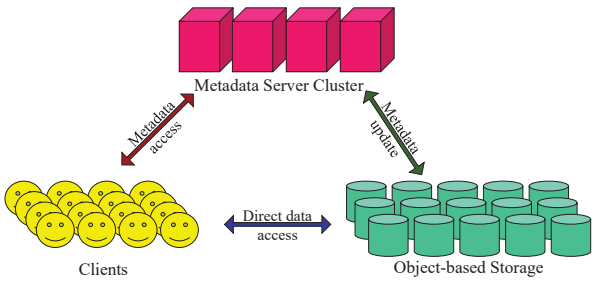

Figure 1: Architecture of distributed file systems.

In this paper, we propose a correlations-based metadata prefetching to reduce IO traffic to MDS, to scale distributed file systems, to increase client cache hit ratio with high accuracy and low false positive. How to define the correlations among files and how to update correlations in real time become a challenge.

2 Background and Related Work

Previous works focus on mining relationships among files according to their access pattern. If two files are frequent accessed closely in the past, they will be accessed together with higher probability near the future. Based on this definition, C-Miner[5] adopts a frequent sequence mining method to explore block correlations. DiskSeen[1] exploits disk layout and based on analysis tof temporal and spatial relationships to improve the sequentiality of disk access.

To reduce the IOs to file system, Kroeger[4] introduces a extended partitioned context modeling(EPCM) to explore correlations by building access trie according to access pattern and prefetches the files with the highest weight. Gu[3] proposes a the weighted-graph-based grouping method to predict and prefetch the following access sequence.

These above two methods require extra space to store the correlations and the algorithm complexity is heavy. Meanwhile, based on the access pattern to exploring correlations is offline, but the correlations among files will evolve as time goes. And they only consider the access pattern and forget the intrinsic relations among files.

3 Our Approach and Novelty

3.1 Correlations Definition

We think that file A and file B are correlated if file A refers to file B or they are accessed closely. In our model, there are two types of correlations which are data correlation and access correlation respectively. If a file refers

or links to another file, for example web file contains the links to scripts, images or other links, these two files are data related. And if two files often access closely, these two files are access related.

Both in data correlation or access correlation, the correlations between files are different. For example, scripts and images are more related than links in web files because the links maybe not clicked by clients but the scripts and images must be loaded when the web files load. Therefore, we introduce a priority to stands for the closer correlations in file system.

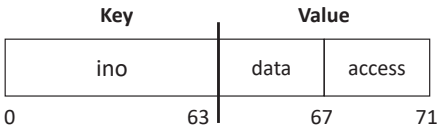

Figure 2: Data structure of correlation.

We propose a key-value structure for correlations as Fig. 2 shows. The Key is the inode number of the correlated file and occupies 8 Bytes. To keep uniqueness of key, inode number should be always increasing. The value is the score of two correlations to make a priority between correlations. The closer correlations are with higher scores.

3.2 Correlations Update

In order to give a better presentation to correlations, we propose three update methods which are online update, offline update and user interface as Fig. 3 shows.

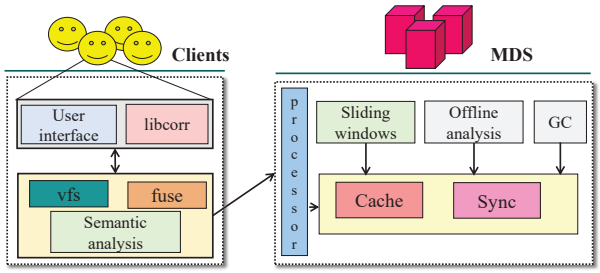

Figure 3: Correlations update.

Online update Semantic analysis and sliding windows procedures are responsible for updating correlations online. When clients update files data, semantic analysis procedure will be trigged to update data correlations in real time. Due to the diverse data formats with multiple types of files, the design of semantic analysis procedure becomes a challenge. For example, scripts, images or other files are linked in web file by `href` or `src`. And source code file includes head file by `include`. Based on the diverse data organization, we propose a aggravate semantic analysis proceeding to file data.

We build a semantic set for different types of files. Each element is responsible for a specific type. When new data is written, the analysis process is triggered. Based on the matching patterns that we pre-defined, this process scans the data to explore the related files and records them. Each correlation is scored by files types and counted to make a difference of correlations.

We create a sliding window on recently access pattern of every client to exploring online access correlations and to exclude concurrency interference simultaneously. The window is with a fixed size and slides ahead as new access sequence comes. The files in window are considered correlated and recorded as *ordered pairs* to keep their access sequence. Each ordered pair has a count and alters dynamically with window slides. Meanwhile, the correlation is higher when access to two files is closer. Hence count and gap are both considered into score.

Offline update Consider that the quality of service, online exploring must be lightweight. To give a better presentation to correlations, we propose a offline method to exploring correlations among files when the system is idle or light-loaded. System will record the access pattern for every client in past which is longer than the online windows and build a access trace to be analysed. To find the correlations among such access traces, a machine learning or deep learning will be adopted. The results are updated in files extended attributes. With offline exploring, more correlations could be found which is not only in single client, but also between clients.

User interface Besides semantic analysis and sliding windows, we also support multiple user interfaces for clients update. All interfaces are supported POSIX interface and need not change clients applications.

- **Set** Clients could set data and access correlations to a specific file with this interface. New score will be embedded into system.
- **Remove** Remove a specific correlation item from a file.

Besides set and remove interfaces, we also support clients **get** a specific correlation from a file or **list** all correlations together.

Challenge 1 When there is overwritten to a file, as Fig. 4 shows, the new data is easy to make a analysis to find correlated files. But the overlayed data maybe contain links to related files, which are obsolete and should be removed simultaneously. However the covered data isn't in cache, so how to update the correlations is a challenge. We introduce a lazy update method. When a files overwrites, we make a `overwritten` tag on the file. Once the tagged file is accessed, system will make a semantic analysis again to rebuild correlations to keep consistency.

Challenge 2 When a file is deleted, the correlations to it which contained in other files becomes obsolete and should be removed at the same time. As Fig. 5 depicts,

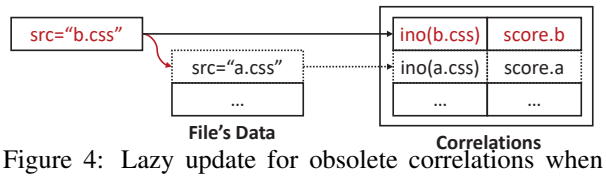

Figure 4: Lazy update for obsolete correlations when deleting files.

file a is deleted. Other four files store a correlations to a. But the correlation is one-direct, system can't remove other correlations to file a right now. To reduce the operation's complexity and overhead, we reserve the obsolete correlations in origin inode. When the file is accessed, the obsolete correlations maybe fetched. Then at that time, these obsolete correlations are found and removed from inode.

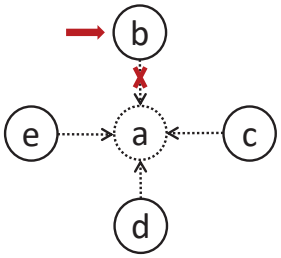

Figure 5: Lazy update for obsolete correlations when deleting files.

3.3 Metadata Prefetching

We propose a aggressive metadata prefetching method to reduce traffic to MDS. In order to increase the hit ratio and reduce space overhead in client, we restrict the number of prefetching items by setting a threshold of correlations scores. When the score of a correlation is higher than the threshold, we think that the correlation is closer and should be fetched together.

To reduce the lookup operations of directory across the network to check the permission, we also batch the ancestors information(directory and inode) with prefetching. System gives the same lease to correlated files as requested file due to the closer correlations. Once clients receive the batched metadata, they insert them into cache to service following requests.

3.4 Implementation and Evaluation

We are building a prototype system based on Ceph, which consists of 32 OSDs and one MDS cluster. We track the IO flows from client to MDS and embed online and offline exploring module into clients and MDS respectively. We are also creating a general library to support user interfaces which could be used for other file systems. We look forward to reducing IOs traffic and memory overhead to MDS and increasing client cache hit ratio via correlations-based metadata prefetching.

4 Conclusion and Future Work

In this paper, we give a presentation of correlations among files from the views of files data and access pattern. We define the correlations and support three methods to update them via online, offline and client-defined. Based on these correlations, we propose a approach to prefetch related metadata to reduce IOs traffic to MDS and increase client cache hit ratio. And we are building a prototype system to evaluate performance.

In the future, we are plan to conduct diverse experiments to compare our approach with other methods on IOs to MDS, client cache hit ratio and memory overhead. Meanwhile, we expect exploring correlations from other views, like clients and namespace to present a extensive view to correlations.

References

[1] X. Ding, S. Jiang, F. Chen, K. Davis, and X. Zhang. Diskseen: Exploiting disk layout and access history to enhance i/o prefetch. In *USENIX Annual Technical Conference*, volume 7, pages 261–274, 2007.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.

[3] P. Gu, J. Wang, Y. Zhu, H. Jiang, and P. Shang. A novel weighted-graph-based grouping algorithm for metadata prefetching. *IEEE Transactions on Computers*, 59(1):1–15, 2010.

[4] T. M. Kroeger and D. D. Long. Design and implementation of a predictive file prefetching algorithm. In *USENIX Annual Technical Conference, General Track*, pages 105–118, 2001.

[5] Z. Li, Z. Chen, S. M. Srinivasan, and Y. Zhou. C-miner: Mining block correlations in storage systems. In *FAST*, volume 4, pages 173–186, 2004.

[6] D. S. Roselli, J. R. Lorch, T. E. Anderson, et al. A Comparison of File System Workloads. In *USENIX annual technical conference, general track*, pages 41–54, 2000.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pages 1–10. IEEE, 2010.

[8] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association, 2006.

[9] S. Zhang, H. Catanese, and A.-I. A. Wang. The Composite-file File System: Decoupling the One-to-One Mapping of Files and Metadata for Better Performance. In *FAST*, pages 15–22, 2016.