Exploring Resource Migration Using the CephFS Metadata Cluster

Michael Sevilla, Scott Brandt, Carlos Maltzahn, Ike Nassi
University of California, Santa Cruz
{msevilla, scott, carlosm, inassi}@soe.ucsc.edu

Sam Fineberg
HP Storage
fineberg@hp.com

1 Introduction

Understanding the effects of migrating resources is an important part of load balancing. Today's systems can already virtualize memory and the ability to migrate other resources, such as CPU, disks, and network, is fast approaching. When we finally have the ability to migrate different resources, how do we know when and where to move them? Such migration will depend on the utilization, configuration, and workload, but how will we weight these factors to design robust, guaranteeable systems? In this work, we propose using metadata management as a substrate for exploring different heuristics for resource migration and load balancing. We hypothesize that an effective metadata management strategy will also depend on the utilization, configuration, and workload.

POSIX-compliant systems are important for legacy software and users accustomed to hierarchical file systems. Unfortunately, file metadata is highly accessed and does not scale for sufficiently large systems in the same way that read and write throughput do [1, 3]. File metadata is very different from regular data; the need to distribute it amongst many nodes is not a result of its size, but its popularity. Maintaining a file system hierarchy and file attributes is notoriously difficult in high-performance computing (HPC), where checkpointing behavior induces "flash crowds" of clients simultaneously opening, writing, and destroying files in the same vicinity (*e.g.*, a directory).

The "big data" era has rendered proven metadata management techniques insufficient for metadata-intensive workloads. For example, Google had to add support for multiple masters to manage metadata because today's workloads often deal with many small files (*e.g.*, log processing) and a large amount of simultaneous clients (*e.g.*, MapReduce jobs) [2]. Suddenly, the metadata problem, once reserved for HPC, has found its way into large data centers.

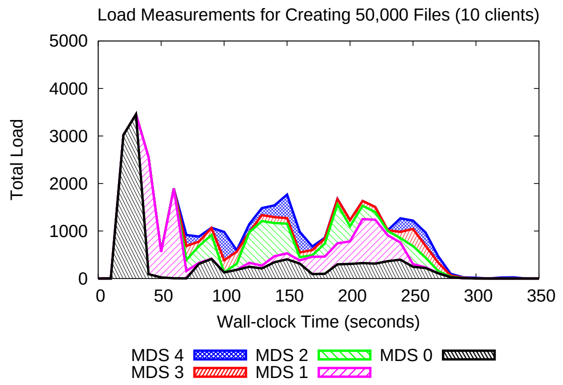
While hash-based metadata management and object stores do well to evenly distribute metadata and its load, they sacrifice the locality inherent in hierarchical file systems. Caching popular inodes can help improve locality, but this technique is limited by the size of the caches and only stores data that has already been seen, instead of data that is related. We use the Ceph file system (CephFS) as a platform for attacking the metadata management problem because it was built with locality in mind and the tools for resource migration and hotspot detection are already implemented.

2 Approach

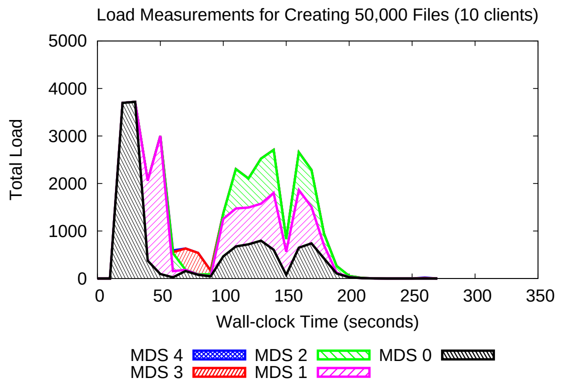
Ceph [3] is a distributed storage platform that stripes and replicates data across a reliable object store (RADOS). Clients talk directly to object storage daemons (OSDs) on individual disks. This is done by calculating the data's placement ("where should I store my data") and location ("where did I store my data") using a hash-based algorithm. CephFS is a POSIX-compliant interface built on RADOS that decouples metadata and data access.

CephFS achieves good locality and equal load distribution in its metadata cluster using dynamic subtree partitioning [4]. Dynamic subtree partitioning migrates subtrees of the file system hierarchy to different metadata servers (MDSs), in response to hotspots and "flash crowds". In CephFS, when many creates/writes are made in the same directory, the metadata contents are fragmented, or partitioned, across multiple MDSs with a hash on the filename. When many reads/opens are made to the same file, the metadata contents are replicated across multiple MDSs. CephFS is purely reactive to a threshold and lacks heuristics for data eviction ("when should I move metadata") and load balancing ("where should I move metadata"). The Ceph engineers have not focused on load distribution and balancing since the original papers [3, 4], so the space is relatively unexplored.

First, we plan to expose the problem by observing how the CephFS metadata cluster reacts to different work-



(a) The metadata cluster can properly balance load.



(b) Switching MDS0 and MDS4 changes migration patterns.

Figure 1: Under the same create-intensive workload, the CephFS metadata cluster performs differently depending on "how" and "when" it migrates resources. (a) shows how CephFS fragments hot directories, while (b) shows that a higher utilization on fewer nodes leads to better performance.

loads. We will model our benchmarks after common use cases for CephFS; talks with the Ceph developers, and the mailing list, indicate that the community is planning to use CephFS as a backup repository, a shared file system, a file server, and/or a compute backend.

Second, we plan to quantify the tradeoffs for migrating resources and dispersing load. We hypothesize that moving subtrees is simple and fast because inodes, which have no allocation metadata, are directly embedded in directories.

Third, we plan to use machine learning techniques to help the system learn about the workload and adapt for the best performance. For example, auto-correlation can pick up on the periodicity of checkpointing and decision trees can predict performance given the metadata layout and the cost of moving a subtree.

3 Current Status

Preliminary experiments show that CephFS can properly balance load amongst the MDSs but the effects on performance and the cluster's behavior are unpredictable. We deployed CephFS with a 5-node metadata cluster and issued 50,000 file create requests in parallel with 10 tasks using `mdtest`, the popular benchmark for HPC. Figure 1 shows a breakdown of the load on the entire cluster over time (x axis), where the total load (y axis) is calculated using the cluster load, individual MDS load, request rate/latency, and CPU utilization.

Figure 1a shows that CephFS can effectively balance load across the metadata cluster. When the load reaches a threshold, MDS0 fragments the hot directory across all 5 MDSs. In Figure 1b, we switch the order of the nodes; the node that used to be MDS4 is now set to be the primary node (MDS0). Although the workload and the involved nodes are the same, the job finishes faster and only uses 3 MDS servers. This indicates that better load balancing does not imply better system performance and that effective resource migration depends on more than just CPU utilization and request rate/latency.

References

[1] Alam, S. R., El-Harake, H. N., Howard, K., Stringfellow, N., and Verzelloni, F. Parallel I/O and the Metadata Wall. In *Proceedings of the 6th Workshop on Parallel Data Storage* (2011), PDSW'11.

[2] McKusick, K., and Quinlan, S. GFS: Evolution on Fast-forward. *Commununications ACM 53*, 3 (Mar. 2010), 42–49.

[3] Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E., and Maltzahn, C. Ceph: A Scalable, High-Performance Distributed File System. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design & Implementation* (2006), OSDI'06.

[4] Weil, S. A., Pollack, K. T., Brandt, S. A., and Miller, E. L. Dynamic Metadata Management for Petabyte-Scale File Systems. In *Proceedings of the 17th ACM/IEEE Conference on Supercomputing* (2004), SC'04.