Master's Thesis
석 사 학 위 논 문

Analyzing Causes of Metadata Service Overheads in Ceph File System

Hojun Kim (김 호 준 金 浩 俊)

Department of Information and Communication Engineering

DGIST

2021

Master's Thesis
석 사 학 위 논 문

Analyzing Causes of Metadata Service Overheads in Ceph File System

Hojun Kim (김 호 준 金 浩 俊)

Department of Information and Communication Engineering

DGIST

2021

Analyzing Causes of Metadata Service Overheads in Ceph File System

Advisor : Professor Sungjin Lee
Co-Advisor: Professor Yeseong Kim

by

Hojun Kim
Department of Information and Communication Engineering
DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics[1)].

June 22, 2021

Approved by

Professor (Advisor) Sungjin Lee (signature)

Professor (Co-Advisor) Yeseong Kim (signature)

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by some-one else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

Analyzing Causes of Metadata Service Overheads in Ceph File System

Hojun Kim

Accepted in partial fulfillment of the requirements for
the degree of Master of Science.

(May 25, 2021)

Head of Committee _____(signature)

Prof. 이 성 진

Committee Member _____(signature)

Prof. 김 예 성

Committee Member _____(signature)

Prof. 좌 훈 승

MS/IC

201922012

김호준. Hojun Kim. Analyzing Causes of Metadata Service Overheads in Ceph File System. Department of Information and Communication Engineering . 2021. 26p. Advisor Prof. Sungjin Lee, Co-Advisor Prof. Yeseong Kim.

Abstract

Distributed File System (DFS) is a popular file system in High-Performance Computing (HPC) due to the demand for a petabyte-scale file system. Among several DFSs, Ceph File System (CephFS) is one of the most widely adopted DFS. It shows advantages in service availability and data reliability. However, CephFS suffers from severe performance degradation when processing requests about a large number of files in HPC environment. The performance degradation is caused by metadata service overheads in CephFS. In this paper, we discovered CephFS metadata service overheads in terms of performance and scalability through metadata performance experiments. Also, we analyzed the causes of overheads by doing additional experiments. The causes of metadata service overheads in CephFS are decoupled metadata service and strict client cache policy in a multi-client environment. We verified the causes of overheads by showing that removing causes of overheads in CephFS improves performance greatly compared to the existing CephFS. Therefore, we expect that this work can help improve the performance degradation of CephFS in the near future.

Keywords: Distributed File System, Ceph File System

List of Contents

List of Tables

List of Figures

I. Introduction

Distributed File System (DFS) is a popular component for computation systems that requires enormous storage space to store data, such as High-Performance Computing (HPC). In HPC environment, applications demand storage capacity of petabytes scale [1]. Ceph File System (CephFS) [2] is a DFS for supporting such applications that require petabyte scale of storage capacity. It is a well-known and one of the most widely adopted DFS in HPC environment. This is because of the advantages of CephFS that it supports not only a large storage capacity but also a high level of service availability and data reliability [3].

However, CephFS suffers from its limited performance when processing requests from a large number of files of HPC workload. This can be a severe problem when adopting CephFS in HPC environment. The poor performance of CephFS when handling a large number of files is originated from its extremely low metadata operation performance and scalability. Measuring file create operation on CephFS using 1 metadata server, which processes metadata operations in CephFS, the throughput of file create operation is revealed as 2375ops/s. Comparing this result to a local file system, Ext4, it is 97.1% lower performance than Ext4. Not only throughput but also latency is slower than a local file system. For a file create operation, it shows 4x longer latency than local file system [4]. However, local file system has an advantage in terms of network overhead. So, we also compared CephFS to Network File System (NFS) which has additional network overhead than Ext4. CephFS showed 50% lower performance in file create operation than NFS which means that CephFS needs much more resources to be scaled as the same performance of other non-distributed file systems like Ext4 or NFS. It can utilize only a small fraction of the entire processing capacity of the storage server. In terms of scalability,

we measured its throughput by adding metadata servers from 1 to 16. It showed that metadata performance is improved only 2x when we increased the number of metadata servers from 1 to 16. It is extremely low scalability considering the resources used for improving performance. However, not all DFSs suffer greatly from their limited scalability. CephFS's scalability is the lowest among many DFSs [5]. These overheads in terms of performance and scalability of CephFS eventually cause a higher cost for DFS service in HPC.

Despite the severity of the metadata service overheads in CephFS, detailed information about overheads and their cause is little known. Therefore, this paper conducted experiments about metadata operations in CephFS to analyze the causes of performance degradation. We measured CephFS performance in detail using diverse experiment setups to find the reasons for the metadata service overhead in CephFS.

As a result, we found two causes of CephFS overhead. The first cause is decoupled metadata service in CephFS. We found that CephFS separates metadata services into two parts, unlike other DFS. Separated parts of the service usually work in a different physical server in the cluster. This incurs additional network overhead when they communicate with each other. To prove this, we configured CephFS cluster to three different setups and measured the performance of each setup. As a result, we found that the cluster configured to have no additional network overhead performs at most 4x better than the other cluster. The second cause is the strict cache policy of CephFS. CephFS doesn't allow multiple clients to cache the metadata objects in the same directory. This makes clients accessing the same directory can't utilize metadata cache and causes huge overhead in the scalability of metadata reading operation. We compared two different experiments, when each client accesses metadata objects located in the same directory and in different directories, to show the performance degradation due to the unavailability of the client cache.

As metadata service performance takes a large portion of the overall performance of DFS, we expect that solving these overheads might help increasing CephFS performance significantly in the future.

II. Background

2.1 Distributed File Systems

DFS is a network file system that can be accessed from different hosts concurrently. It distributes data and metadata of a file to the storage servers for availability and scalability. Hosts can handle files in DFS just like the files in local file systems. Commonly used DFS are GlusterFS, Lustre, CephFS.

DFS's metadata storing method can be categorized in two ways, hash-based store and directory-based store. Hash-based store determines the location of file's metadata by hashing their unique id value such as file's path. Thus, this method can spread metadata throughout the server cluster evenly. However, hashing places metadata without considering the file's location in a directory hierarchy. No matter how close two files are in a directory tree, it places the metadata in a random server. Therefore, it requires multiple servers to be involved in processing metadata requests about files in the same directory. GlusterFS and Lustre use hash-based store to distribute metadata. Directory-based store considers file's location in a directory hierarchy. It places files that have a similar path to the same server. However, it does not evenly distribute metadata to the storage servers. There is a chance that metadata requests concentrate on part of the servers if certain directories contain most of the files. CephFS uses directory-based store to distribute metadata.

2.2 RADOS Storage Cluster

Ceph is a set of distributed object storage services that provides 3 kinds of different storage services, file store, block store, and object store [6]. The file storage service is also called CephFS. Ceph stores different types of data with high availability and strong fault tolerance for a large amount of data. Data distribution and management through multiple servers are taken by RADOS [7] using an efficient CRUSH algorithm [8]. Each of Ceph storage services converts a given type of data such as a file, block, or object to RADOS object and RADOS stores objects safely. Ceph

RADOS service consists of 4 types of daemons. Monitor daemon stores a master copy of the entire cluster map which is used for other daemons to cooperate with each other. Manager daemon deals with an overall system's runtime status for administrators to manage the cluster. Object Storage Device (OSD) daemon records data objects to a storage device and replicates objects to other OSD daemon to provide fault tolerance and high availability. Metadata Server (MDS) daemon manages the processing of file metadata to alleviate the load of other daemons. MDS daemon is required when CephFS, Ceph's file storage service, is used. Each of these daemons can be installed on multiple servers to maintain a storage service even in a situation of single or several server failures in the cluster.

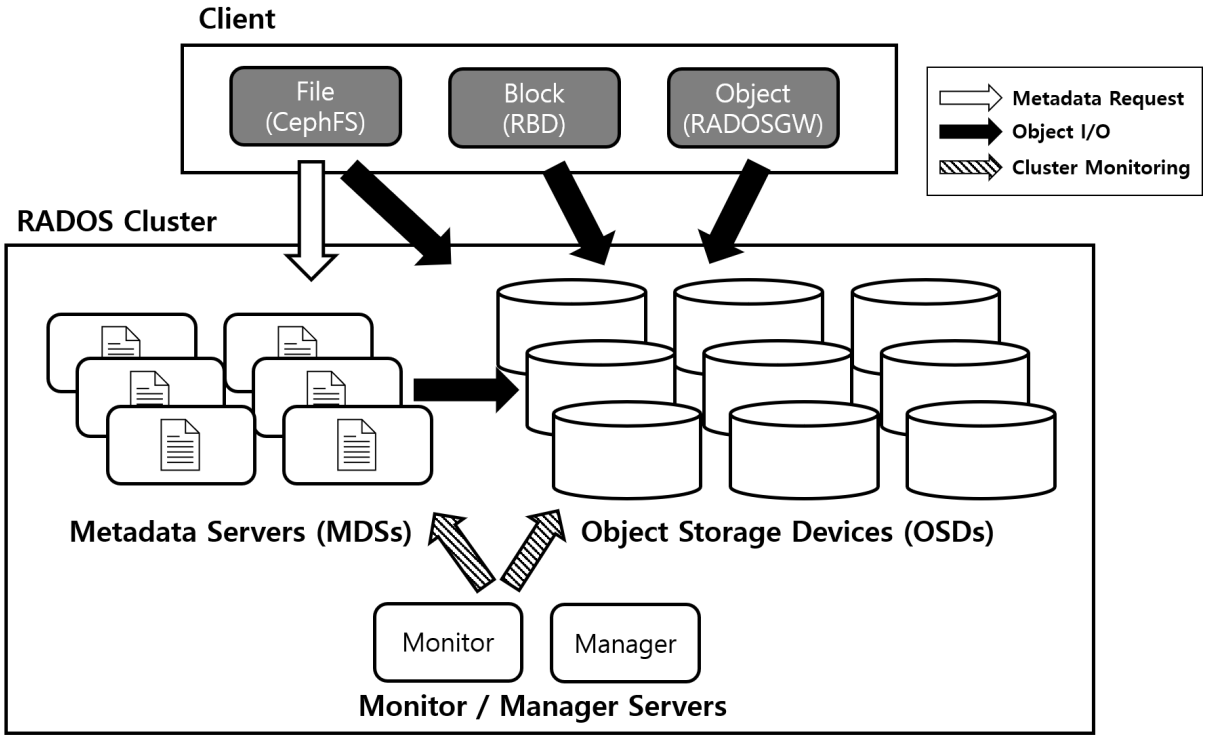


Figure II.1: Architecture of RADOS Storage Cluster

2.3 Ceph File System

CephFS is a file storage service that runs on top of RADOS cluster. CephFS stores file's metadata and data to OSDs in RADOS cluster. In the case of metadata, clients send a metadata request to MDS daemon. MDS daemon handles metadata operations received from clients and stores/retrieves metadata from OSDs. Then, it sends metadata to the clients. In the case of data, after clients got metadata from MDS, they directly access to OSDs to read/write file's data. CephFS distributes metadata to multiple MDS daemons in the cluster. It supports each MDS to migrate the management of metadata to other MDSs to distribute metadata traffic evenly.

MDS takes charge of processing metadata in certain areas of a directory tree. This area is determined by the algorithm called dynamic subtree partitioning [2]. Dynamic subtree partitioning basically uses directory-based distribution. But it relieves the disadvantage of directory-based distribution by dynamically migrate metadata. Dynamic subtree partitioning checks the metadata load of each MDS daemon and adjusts the imbalance by migrating the management of metadata to other MDS. Specifically, when a branch of the directory tree takes more metadata load than other branches, the MDS daemon that manages the branch handles more requests than the other, causing overhead in scalability. Thus, CephFS detects such branches and migrates the branches to another MDS daemon. By doing so, it distributes metadata traffic evenly to MDS servers and helps improve the metadata service scalability of CephFS.

To reduce unnecessary requests to RADOS cluster, CephFS allows each client and MDS to cache file's data and metadata [2]. However, caching is more complex than a local file system due to the existence of multiple clients. In a multi-client environment, CephFS should consider cache coherence throughout clients. Because of this, when a client access files that are already cached in another client, MDS should ask the client for eviction of cache and return the

modified version of metadata. By doing so, CephFS can ensure clients access to an up-to-date version of file's data and metadata.

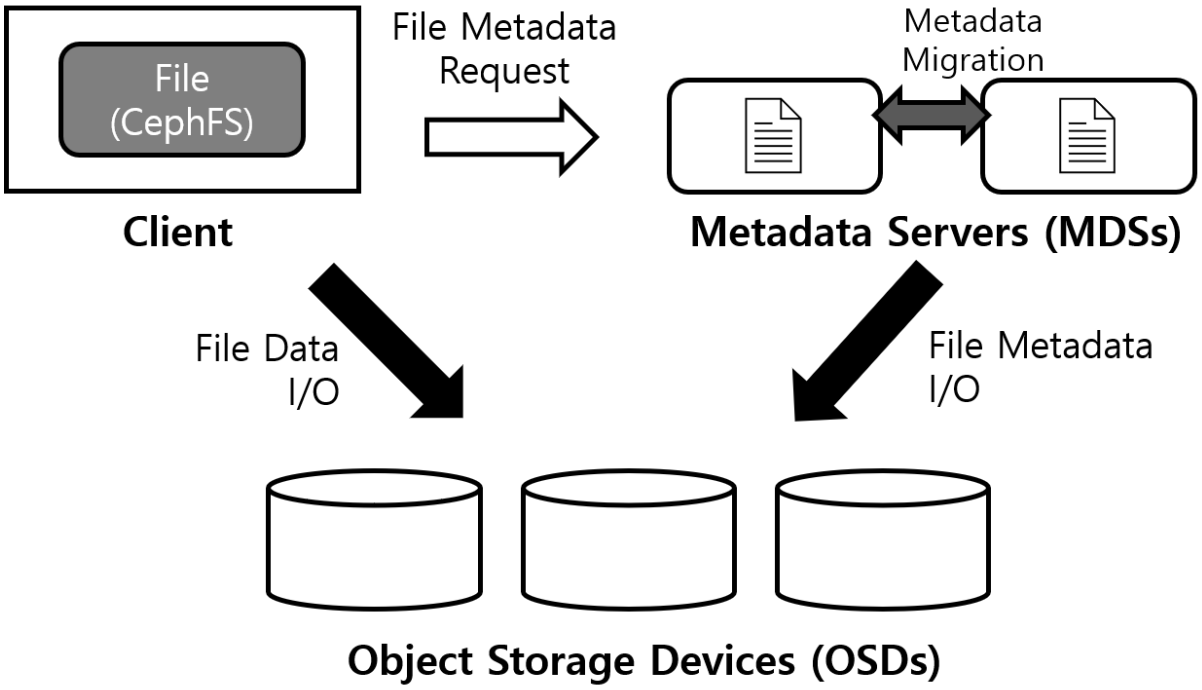


Figure II.2: Architecture of Ceph File System

III. Motivation

To examine CephFS metadata service overhead, we did several performance experiments about CephFS. Experiments were conducted about 3 kinds of metadata operations that are create, remove, and stat. Each operation represents metadata create, metadata remove, metadata read. As a benchmark tool, mdtest [9] was used for generating metadata requests. Mdtest supports benchmark about various types of file operations over millions of files. 1 million files were used for every experiment.

3.1 Limited Performance Scalability of CephFS

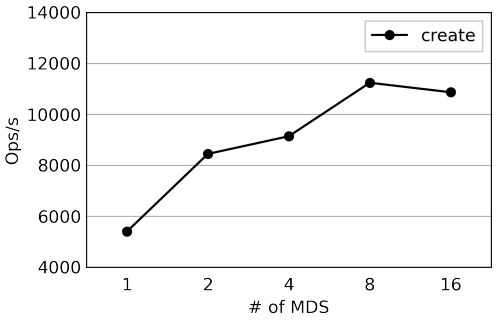
To benchmark the scalability of MDS, we measured each metadata operation's throughput by increasing the number of MDS servers from 1 to 16.

Figure III.1 represents throughput for file create, file remove and file stat with varying number of MDS. In Figure III.1, all metadata operations throughput showed insufficient scalability. Create and remove operation throughput is improved only 108.2%, 46.6%, and 243.8% even if the number of MDS increased from 1 to 16. This means that adding additional MDS daemons doesn't improve the performance of CephFS metadata operations linearly. Furthermore, in some cases, performance was decreased when we added more MDS daemons to the cluster.

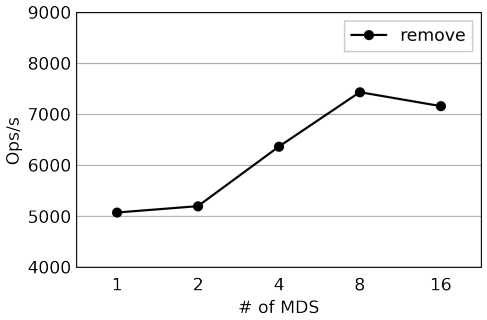
The performance of CephFS also does not scale with the number of clients. We measured file stat throughput by increasing the number of clients from 1 to 7. Figure III.2 shows file stat performance of CephFS when adding clients. When the number of clients exceeds two, the performance of file stat drops 98% which is a significant problem in reading files in a multi-

client environment.

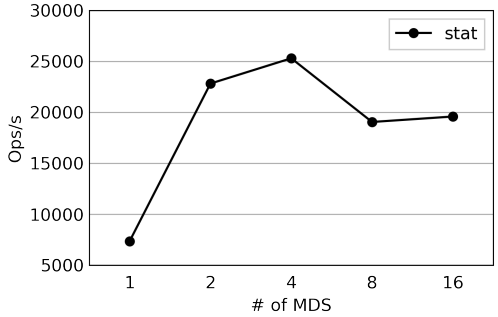
All of these results mean that there is a critical problem in several metadata operations in terms of scalability. Because of this, among many DFSs, CephFS shows the lowest scalability. To compare GlusterFS and Lustre with CephFS, CephFS showed about 90% lower performance scalability than other file systems in file create, file remove, directory create and directory remove operations [5].



(a) Create Operation



(b) Remove Operation



(c) Stat Operation

Figure III.1: Metadata Performance of CephFS with varying number of MDS

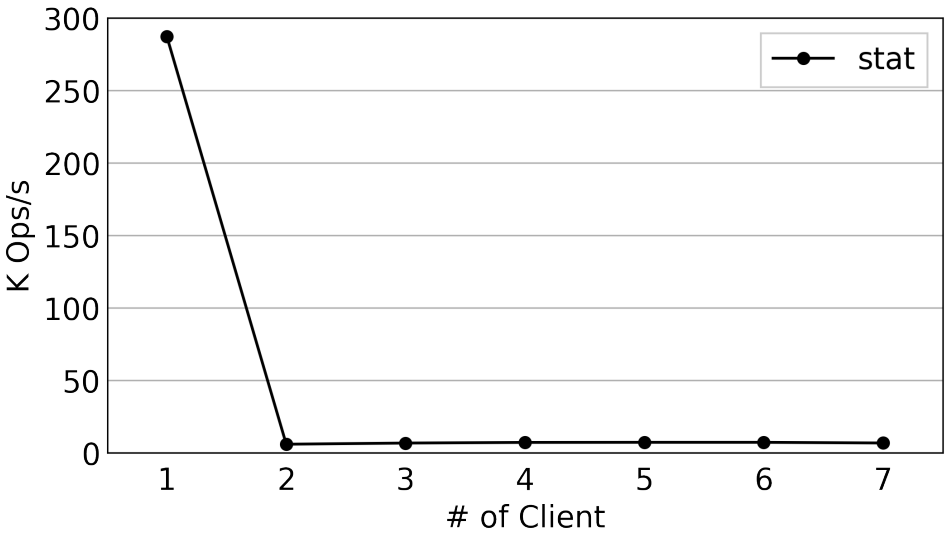
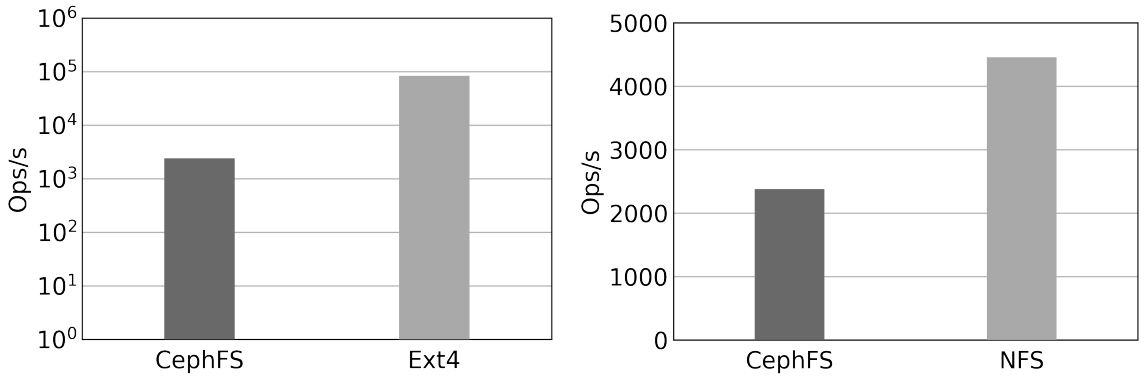


Figure III.2: Metadata Performance of CephFS with varying number of Clients

3.2 Low Single MDS Performance

Not only scalability but also single MDS of CephFS shows poor performance compared to other file systems. To speculate it, we compared CephFS which has one MDS daemon to Ext4 local file system and NFS.

Figure III.3 shows file create throughput comparison among CephFS, Ext4 and NFS. In Figure III.3, CephFS using single MDS showed only 2375 ops/s in file creation when Ext4 is measured as 83328 ops/s. Due to the CephFS's disadvantage in network latency compared to the local file system, there is a huge performance gap between local file system and CephFS. However, NFS also showed higher performance than CephFS although both file systems are accessed remotely by clients through the network. Therefore, CephFS is behind other DFS in terms of scalability and is behind other non-distributed file systems in terms of throughput, requiring much more resources than other file systems.



(a) Comparing CephFS to Ext4

(b) Comparing CephFS to NFS

Figure III.3: Performance Comparison of Single MDS CephFS, Ext4 and NFS

IV. Performance Analysis

By doing a series of experiments, we discovered two causes of CephFS performance overhead that are related to CephFS's internal problems. In this section, we analyzed each experiment result and cause of the metadata performance overhead in CephFS.

4.1 Experiment Environment

The experiments were conducted using servers that have different specifications and usage. There are Monitor, OSD, MDS, Client servers with the following specifications.

	Monitor	**OSD**	**MDS**	**Client**
CPU	Xeon E5-2690 x2	Xeon E5-2630 v2 x2	Xeon E3-1240 v3 x1	Xeon E5-2630 v3 x2
Memory	64GB	80GB	32GB	256GB
Storage	480GB SSD	6TB HDD * 12 1TB SSD * 2	480GB SSD	240GB SSD
Network	10GbE	40GbE	1GbE	10GbE

table IV.1: Server Specification

For experiments, Monitor server is configured with 2 Xeon E5-2690 CPUs, 64GB of memory, 480GB of SSD, and 10Gb ethernet network. We always used a single Monitor server. OSD server, which stores data, is configured with 2 Xeon E5-2630 v2 CPUs, 80GB of memory, 72TB storage capacity of HDD, 2TB storage capacity of NVMe SSD, and 40Gb ethernet network. HDD stores the object of CephFS persistently, while SSD is used for write-ahead logging for Ceph. The network is set to 40Gb to fully utilize the bandwidth of NVMe SSD. We used 4 OSD servers and performed 3-copy replication for data objects of CephFS. MDS server con-

sists of one E3-1240 v3 CPU, 32GB of memory, 480GB of SSD, and 1Gb Ethernet network. There are 16 MDS servers in our cluster but we adjusted the number of MDS servers in each experiment to make environments that correspond to the purpose of the experiment. Client server is configured with 2 Xeon E5-2630 v3 CPUs, 256GB of memory, 240GB of SSD, and 10Gb ethernet network. It has the most powerful computation power among the other servers in order to send a large number of metadata requests to the CephFS cluster. There are 7 client servers that are used for experiments. Similar to MDS servers, we adjusted the number of client servers according to the purpose of the experiments.

We used Ubuntu 18.04 as an operating system for every server. Ceph version 15.2.8(Octopus) is used for CephFS cluster deployment. For benchmark, we used mdtest [9] which is a tool for measuring the throughput of metadata operations such as directory create, directory remove, file create, file remove, and file metadata read. Mdtest supports OpenMPI which enables to control of multiple client servers to send metadata requests to CephFS concurrently.

4.2 Decoupled Metadata Service

RADOS cluster has 4 different types of services that compose the system. Monitor, Manager, OSD services are the main services of the RADOS. MDS service is an additional service that is needed to run CephFS on RADOS. MDS processes metadata requests from clients and sends metadata objects to OSDs. Storing metadata objects is conducted by OSD service. Therefore, storing part and processing part of the metadata service run in a separated machine in CephFS. Each part has to communicate with the other through the network. However, other DFSs have simpler steps to process and store metadata. In other DFSs, such as Lustre or Gluster, storing and processing metadata are combined as a single service. Thus, if a file is newly created by a client, other DFSs usually process and store the file's metadata in a sin-

gle physical server without communicating to other servers. However, CephFS requires more than one server to communicate with each other for handling metadata requests. This incurs performance degradation due to the latency of transferring data through a network.

To verify performance degradation from decoupled metadata service, we conducted the following experiments with different cluster setup same as figure IV.1.

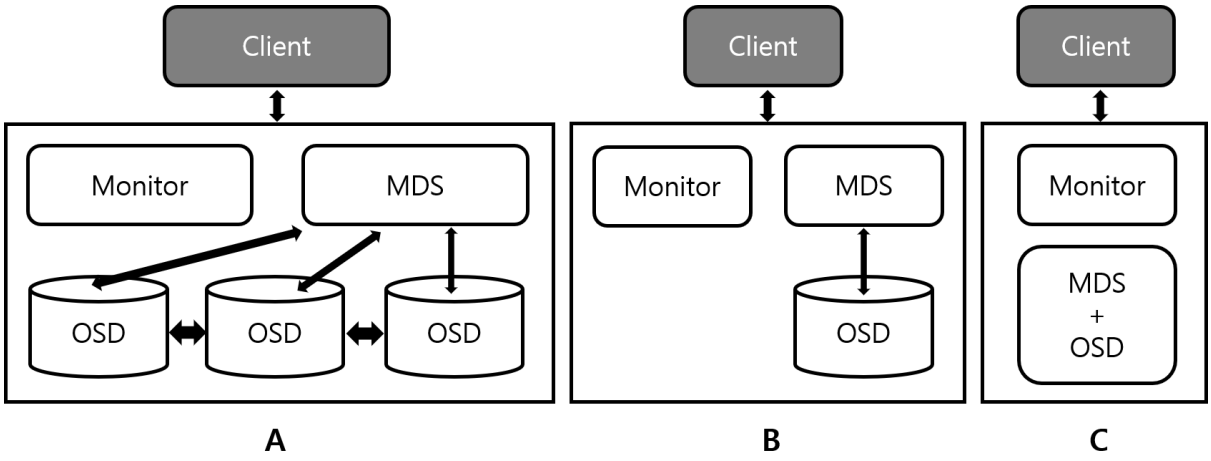


Figure IV.1: Structure Setups of CephFS Cluster

We made cluster A that is configured with 1 client server, 1 MDS server, and 3 OSD servers connected to each other through the network. This cluster stores metadata objects with 3-copy replication. Each copy is stored in a different OSD server for service availability and data reliability. Cluster B is configured with 1 client server, 1 MDS server, and 1 OSD server. This cluster stores data with just one copy. Last cluster C is also configured with 1 client server, 1 MDS server, and 1 OSD server. But this cluster uses MDS and OSD that is installed in the same physical server for reducing additional network overhead between MDS and OSD.

In Figure IV.2, the experiment result showed that cluster A and cluster B have no performance difference in every metadata operation while cluster C showed up to 4x performance improvement. It means that network latency between MDS and OSD is a major overhead for CephFS than data replication.

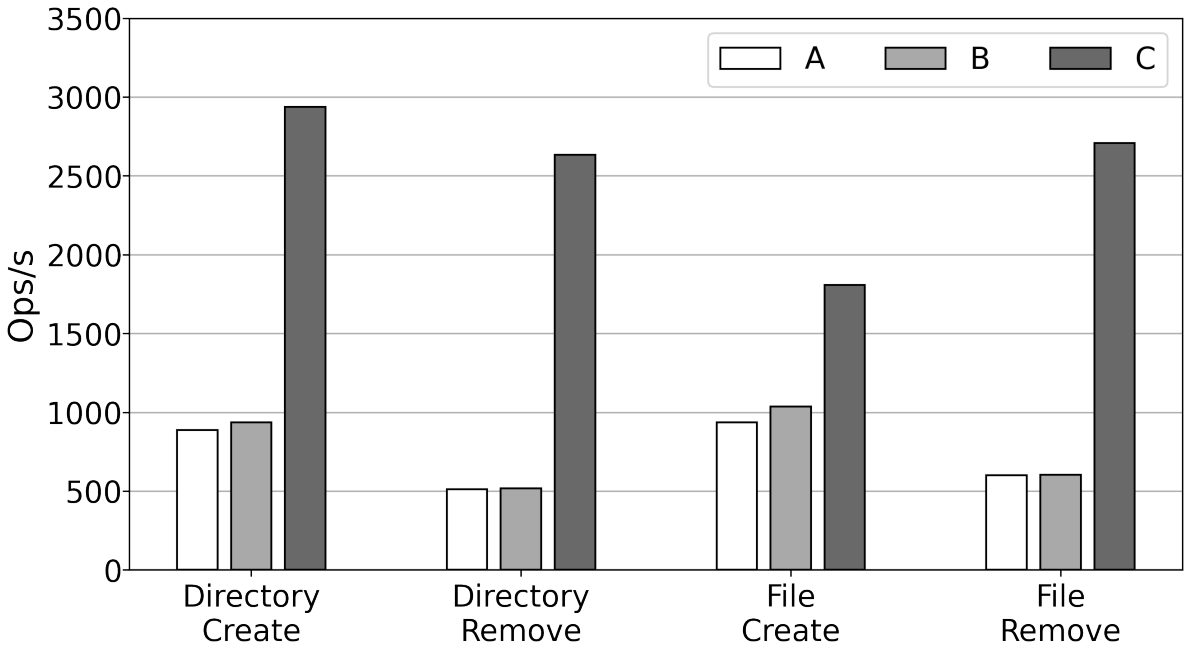


Figure IV.2: Performance Comparison in Different CephFS Cluster Configuration

4.3 Strict Client Cache Policy

Cache policy is another cause of performance overhead in CephFS. Due to its strict cache policy, CephFS suffers from performance degradation while processing metadata read operations. When a client accesses metadata, CephFS caches it into the client's cache. It helps to reduce the effort to read metadata from the CephFS cluster every time. However, when another client accesses the metadata in the same directory, the client, which has cached the metadata, evicts it and returns the modified metadata to MDS. Therefore, if metadata objects in a single directory are accessed by multiple clients concurrently, clients can't cache the metadata. To check this, we conducted the following experiment.

We compared CephFS metadata read performance when clients access the same directory and when clients access different directories. In the first experiment, all clients accessed a single directory and performed file metadata read in that directory. Each client reads different file's metadata in the same directory. In the second experiment, clients accessed different directories and performed file metadata read in those directories. In both experiments, we used a single MDS server and varied the number of clients from 1 to 7 to observe whether performance scales as the client increases.

Figure IV.3 represents file stat performance when each client accesses the same directory or different directories. In Figure IV.3, when clients access the same directory, it showed significant performance degradation if there are more than 2 clients accessing the directory. Metadata read throughput dropped over 90% when the number of clients increased from 1 to 2. However, when clients read metadata from different directories, CephFS showed no performance degradation and the throughput increased linearly as adding more clients. Therefore, a strict cache policy of CephFS can be a severe problem especially when clients access the same directory.

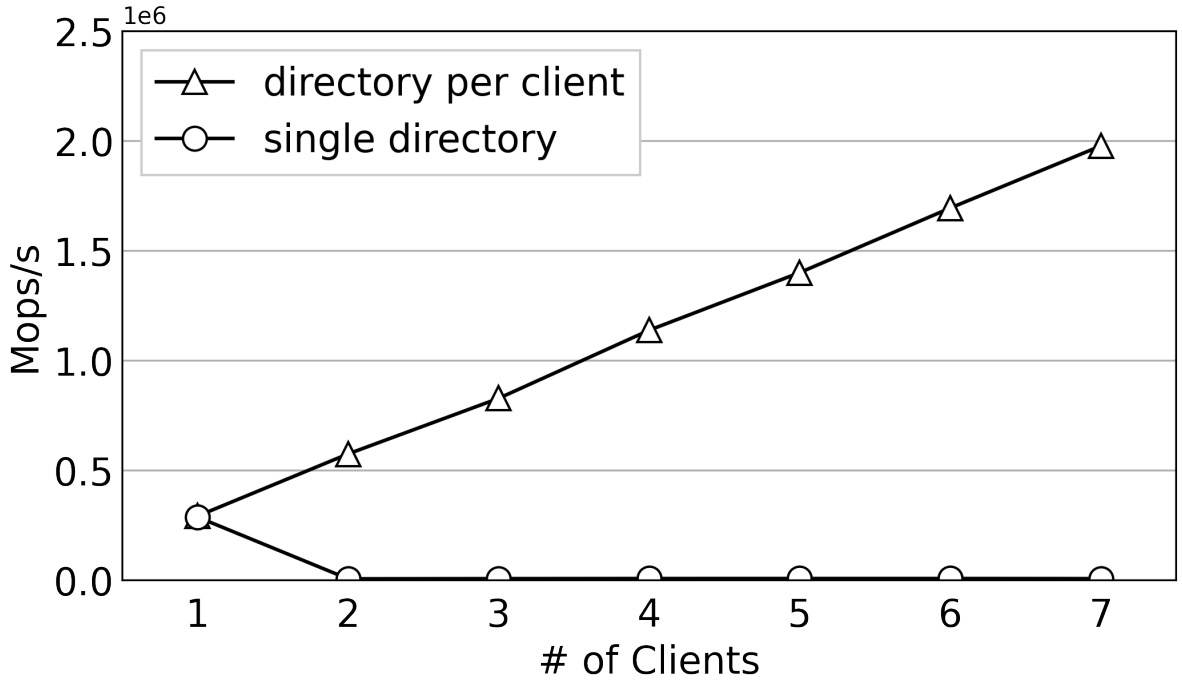


Figure IV.3: Performance Comparison when Clients Access Same/Different Directory

V. Discussion

5.1 Solutions for Metadata Service Overheads

In this section, we discuss the solutions of CephFS overheads and their impact. To improve the low performance of single MDS, we need to merge a decoupled metadata service of MDS and OSD into a single service. Considering that CephFS is one of the services of RADOS and MDS is an auxiliary component for supporting CephFS, modifying OSD for CephFS service isn't appropriate as a solution. To constrain the impact of modification within CephFS, we need to make changes within MDS service. By adding metadata storing functionality to MDS service, we can prevent additional network overhead between MDS and OSD service. However, maintaining the same level of service availability and data reliability in metadata storing functionality requires considerable engineering effort for CephFS. Therefore, we must use the existing metadata storing service of OSD for high availability and reliability while utilizing the metadata storing functionality in MDS for a faster response of metadata operations.

Another problem is a strict cache policy which incurs low scalability of CephFS according to the number of clients. There are many solutions for solving this problem. One of the solutions is conducting cache eviction for single file metadata throughout clients only when other clients update the metadata. The two kinds of problems of strict cache policy are that cache eviction is caused by reading metadata and is caused by accessing other metadata objects in the same directory. By restricting the impact of cache eviction to a single file's metadata that is being updated, we can solve the performance scalability overhead. However, the implemented solution can cause another overhead because we have to alarm all clients for every cache eviction if we don't know the list of clients that are caching an older version of metadata. Therefore,

it is expected that CephFS will need an additional data structure for each metadata to manage a list of clients that are caching the metadata.

VI. Related Works

Although not specific to CephFS, studies on the degradation of metadata performance of other DFS including CephFS have been actively conducted in the past [10, 11, 5]. They found the causes of metadata performance degradation in DFS and suggested new techniques for performance improvement.

GIGA+ is a concept for supporting scalable directories in terms of the number of files in DFS. DFS shows good performance when managing a small number of huge files. However, it is not good at managing a large number of files in a single directory. This kind of weakness can be also found in our experiment about cache policy in CephFS. To overcome such weakness, the authors proposed to divide a directory of DFS into multiple partitions that can be spread to multiple servers. By comparing GIGA+ and CephFS, the authors showed that GIGA+ performs over 3x better than CephFS in terms of file creation performance scalability.

IndexFS adopted GIGA+ techniques to make a fully functional file system that runs as middleware on other DFS without any modification of DFS. IndexFS uses metadata distribution and caching technique of GIGA+ for scalability. And it uses LevelDB to store metadata objects together as a large file in a DFS. Together with GIGA+ technique and LevelDB backend, IndexFS achieved scalability over twice of GIGA+ with showing almost linear scalability.

LocoFS proposed loosely-coupled metadata service to reduce a performance gap between metadata service and key-value store in a file system such as IndexFS. The authors designed three different techniques. They decoupled file metadata service and directory metadata service for better metadata traversal latency. Also, they made metadata objects to be placed on a flat space by eliminating dependencies about directory inode for improved throughput. And they

divided a metadata object into access part and content part to store metadata object in a key-value store friendly way. Consequently, LocoFS achieved better scalability by showing over 3x performance than IndexFS in terms of file creation.

VII. Conclusions

Although CephFS is a widely adopted DFS, it shows low performance and scalability when processing metadata operations about a large number of files. These characteristics of CephFS hinder CephFS cluster to operate efficiently when handling a large number of clients and files. Therefore, we analyzed reasons for metadata operation performance overhead in CephFS by conducting a series of experiments.

We verified that there are two causes of metadata performance overhead in CephFS. The first cause is metadata service that is separated into two parts, processing part and storing part. This decoupled service incurs extra network overhead between two parts. We showed that the metadata performance of CephFS can be improved at most 4x when we removed the extra network overhead. The second cause is a strict client cache policy in a multi-client environment. When clients access metadata in the same directory, CephFS doesn't allow clients to cache metadata due to its strict cache policy. To verify this problem, we measured the performance of CephFS when each client access the same directory or different directories. Then we showed metadata read operation is faster when each of the clients accesses different directories.

We expect that the causes of CephFS metadata service overhead we found will be used for future CephFS improvements and help increase the efficiency of CephFS metadata service.

References

[1] J. Blomer, "A survey on distributed file system technology," in *Journal of Physics: Conference Series*, vol. 608, no. 1. IOP Publishing, 2015, p. 012039.

[2] 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*, 2006, pp. 307–320.

[3] "Ceph user survey 2021," accessed on: 2021-06-21. [Online]. Available: https://ceph.io/wp-content/uploads/2021/05/Ceph-User-Survey-2021.pdf

[4] M. Pillai, "Exploring the performance limits of cephfs in nautilus," May 2019, accessed on: 2021-06-21. [Online]. Available: https://youtu.be/UtJvAWRsj9I

[5] S. Li, Y. Lu, J. Shu, Y. Hu, and T. Li, "Locofs: A loosely-coupled metadata service for distributed file systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[6] "Ceph architecture," accessed on: 2021-06-21. [Online]. Available: https://docs.ceph.com/en/latest/architecture/

[7] S. A. Weil, A. W. Leung, S. A. Brandt, and C. Maltzahn, "Rados: a scalable, reliable storage service for petabyte-scale storage clusters," in *Proceedings of the 2nd international workshop on Petascale data storage: held in conjunction with Supercomputing'07*, 2007, pp. 35–44.

[8] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "Crush: Controlled, scalable, decentralized placement of replicated data," in *SC'06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*. IEEE, 2006, pp. 31–31.

[9] "Hpc io benchmark repository (ior)," accessed on: 2021-06-21. [Online]. Available: https://github.com/hpc/ior

[10] S. Patil and G. A. Gibson, "Scale and concurrency of giga+: File system directories with millions of files." in *FAST*, vol. 11, 2011, pp. 13–13.

[11] K. Ren, Q. Zheng, S. Patil, and G. Gibson, "Indexfs: Scaling file system metadata performance with stateless caching and bulk insertion," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 237–248.

요 약 문

Ceph 파일 시스템의 메타데이터 성능 오버헤드의 원인에 대한 분석

최근 고성능 컴퓨팅과 같은 분야에서 페타바이트 크기의 파일 시스템을 요구하면서 분산 파일 시스템은 고성능 컴퓨팅 분야에서 널리 쓰이는 파일시스템이 되었다. 그 중에서도 Ceph 파일 시스템은 서비스 가용성과 데이터 안정성 덕분에 가장 널리 쓰이는 분산 파일 시스템 중 하나이다. 하지만 Ceph 파일 시스템은 고성능 컴퓨팅 환경에서의 많은 수의 파일에 대한 요청을 처리함에 있어 심각한 성능 저하가 발생한다는 문제점이 있다. 이는 Ceph 파일 시스템에 존재하는 메타데이터 서비스 오버헤드에 의한 성능 저하이며 이 논문에서는 메타데이터 성능 실험을 통해 Ceph 파일 시스템의 성능 측면과 확장성 측면에서 메타데이터 서비스 오버헤드가 존재함을 발견하였다. 또한 추가적인 실험을 통해 이 오버헤드들의 원인을 분석하였으며 분리된 메타데이터 서비스와 다중 클라이언트 환경에서의 엄격한 클라이언트 캐시 정책이 오버헤드의 원인인 것으로 나타났다. 우리는 Ceph 파일 시스템에서 오버헤드의 원인들을 제거했을 때 성능이 기존의 Ceph 파일 시스템보다 크게 개선됨을 보여줌으로써 오버헤드의 원인을 증명하였으며 이 논문의 결과가 추후에 Ceph 파일 시스템의 성능 저하를 개선하는데 있어 도움이 될 것이라 기대한다.

핵심어: 분산 파일 시스템, Ceph 파일 시스템