


Article

Supporting SLA via Adaptive Mapping and Heterogeneous Storage Devices in Ceph[†]

Sopanhapich Chum¹, Heekwon Park² and Jongmoo Choi^{1,*} 

¹ Department of Computer Science and Engineering, Dankook University, Yongin 16890, Korea; 72191829@dankook.ac.kr

² Memory Solutions Lab., Samsung Semiconductor Inc., San Jose, CA 95134, USA; heekwon.p@samsung.com

* Correspondence: choijm@dankook.ac.kr; Tel.: +82-31-8005-3242

[†] This article is an extended version of the paper “SLA-Aware Adaptive Mapping Scheme in Bigdata Distributed Storage Systems” presented in the 9th International Conference on Smart Media and Applications (SMA 2020), Jeju, Korea, 17–19 September 2020.

Abstract: This paper proposes a new resource management scheme that supports SLA (Service-Level Agreement) in a bigdata distributed storage system. Basically, it makes use of two mapping modes, isolated mode and shared mode, in an adaptive manner. In specific, to ensure different QoS (Quality of Service) requirements among clients, it isolates storage devices so that urgent clients are not interfered by normal clients. When there is no urgent client, it switches to the shared mode so that normal clients can access all storage devices, thus achieving full performance. To provide this adaptability effectively, it devises two techniques, called logical cluster and normal inclusion. In addition, this paper explores how to exploit heterogeneous storage devices, HDDs (Hard Disk Drives) and SSDs (Solid State Drives), to support SLA. It examines two use cases and observes that separating data and metadata into different devices gives a positive impact on the performance per cost ratio. Real implementation-based evaluation results show that this proposal can satisfy the requirements of diverse clients and can provide better performance compared with a fixed mapping-based scheme.

Keywords: distributed storage system; service level agreement; adaptive mapping; heterogeneous devices



Citation: Chum, S.; Park, H.; Choi, J. Supporting SLA via Adaptive Mapping and Heterogeneous Storage Devices in Ceph. *Electronics* **2021**, *10*, 847. <https://doi.org/10.3390/electronics10070847>

Academic Editor: Juan M. Corchado

Received: 23 February 2021

Accepted: 30 March 2021

Published: 2 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Bigdata analysis is a vital ingredient in modern internet services such as e-commerce, semantic search, and customer recommendation [1–3]. In addition, deep learning algorithms rely on bigdata to improve their classification and inference capability [4–6]. Several applications such as daily trading, traffic analysis, and smart infrastructure require a predictable bigdata analytic framework [7–10]. According to IDC, world data will grow from 33 ZB (ZettaByte) in 2018 to 175 ZB by 2025, with a compounded annual growth rate of 61% [11].

To manage bigdata in a scalable and reliable manner, a lot of distributed storage systems are being developed actively both in the industry and academia. Typical examples include GFS [12], HDFS [13], Ceph [14], Azure Storage [15], Amazon S3 [16], Openstack Swift [17], Haystack [18], Lustre [19], GlusterFS [20], and so on. They have a scalable mapping mechanism to determine in which storage devices client data reside. In addition, they make use of a replication technique or erasure code for high reliability and fast recovery.

As the usage of distributed storage systems expands, the demand for supporting different QoS (Quality of Service) requirements among clients is also increasing. For instance, an intelligent transportation system needs to monitor traffic in a timely manner, which in turn requires accessing data within a predefined time constraint [21,22]. On the contrary, backup or batch processing clients can access data in a best-effort way without interrupting urgent clients [23,24]. Such different requirements are contracted as SLA

(Service-Level Agreement) between clients and storage providers, which can be expressed in various forms such as performance, reliability, and cost [25–28].

This paper proposes a novel resource management scheme for supporting SLA in a distributed storage system. It classifies clients into two types: one is urgent clients who require guaranteed performance, and the other is normal clients who can be served in a best-effort way. One straightforward scheme is dividing storage devices into two regions, where one region is mapped into urgent clients while the other region is mapped into normal clients. This isolation prevents normal clients from interfering with urgent clients. However, this fixed mapping has a potential to degrade performance especially when the activity of urgent clients becomes low.

To overcome this performance degradation, the proposed scheme makes use of not only the isolated mode but also the shared mode in an adaptive manner. In the isolated mode, storage devices are divided between urgent and normal clients to differentiate clients according to their SLAs. When the activity of urgent clients decreases below a predefined threshold, it switches to the shared mode, allowing normal clients to use all storage devices, thus achieving full performance.

In order to make this adaptation seamlessly, it devises two techniques, called *logical cluster* and *normal inclusion*. The logical cluster is designed to provide different abstractions of a distributed storage system depending on the modes. The normal inclusion is a type of property that ensures the accessibility of data in the isolated mode that have been written in the shared mode.

In addition, this paper examines how to exploit heterogeneous storage devices, HDDs (Hard Disk Drives) and SSDs (Solid State Drives), to support SLA. It reveals that a heterogeneity-oblivious mapping does not reap the performance benefit of SSDs due to the replication mechanism in a distributed storage system. To address this problem, this paper investigates two use cases. One is employing heterogeneous storage devices for different client types, called *normal/urgent separation*. The other is taking a different approach, called *data/metadata separation*, that uses SSDs for metadata only to obtain both the capacity benefit of HDDs and the performance benefit of SSDs simultaneously.

The proposal was implemented in Ceph that is a widely used open-source-based distributed storage system that deploys various cloud platforms including Hadoop, Amazon S3, and Openstack [29]. Specifically, the CRUSH (Controlled Replication Under Scalable Hashing) algorithm [30] is modified to materialize our proposal. The evaluation results show that the proposed scheme can guarantee SLA required by urgent clients while enhancing performance by up to 18% compared with a fixed mapping scheme. It also exhibits that the data/metadata separation is a cost-effective solution for heterogeneous storage devices.

The contributions of this paper can be summarized as follows.

- It explores diverse design spaces to ensure SLA in a distributed storage system from three perspectives: mapping mechanism, client type, and storage heterogeneity.
- It proposes an adaptive mapping based on two new techniques, called logical cluster and normal inclusion, to provide different abstractions and to assure data accessibility, respectively.
- It examines two use cases, called normal/urgent separation and data/metadata separation, for heterogeneous storage devices with the consideration of size and access pattern.
- It provides real-implementation-based experimental results in terms of QoS, performance, and cost.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation of this study. Section 3 explains our proposal. The experimental environment and evaluation results are discussed in Section 4. Related works are surveyed in Section 5. Finally, the conclusion and future work are presented in Section 6.

2. Background

This section first explains Ceph in detail. Then, it discusses two observations that motivate this study.

2.1. Ceph Architecture

Ceph is a scalable, reliable, and open-source distributed storage solution [14,31]. It is popularly used as backend storage for cloud computing including Openstack Swift, Amazon S3, and Redhat OaaS (Object storage as a Service) [32]. It is also deployed for bigdata and deep learning platforms such as Hadoop and Tensorflow [33–35].

Figure 1 presents the logical structure of Ceph. It consists of four layers: the client, interface, RADOS (Reliable Autonomic Distributed Object Store), and storage layers. Ceph provides three interfaces: (1) RGW (RADOS Gateway) for an object storage, (2) RBD (RADOS Block Device) for a virtual block device, and (3) CephFS for a distributed file system with POSIX semantics.

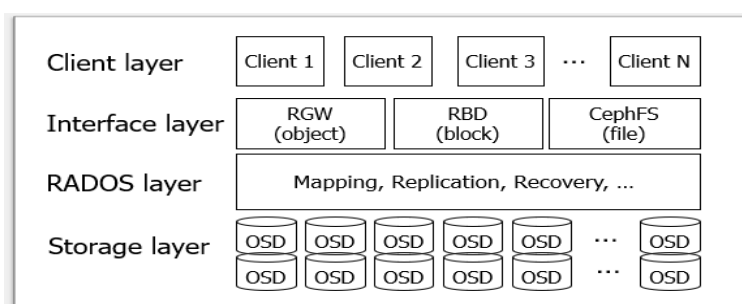


Figure 1. Ceph structure, which consists of four layers: client, interface, RADOS (Reliable Autonomic Distributed Object Store), and OSDs (Object Storage Daemons).

RADOS is an essential layer in Ceph [36]. It takes charge of mapping between clients and OSDs (Object Storage Daemons), which will be discussed further in Section 2.2. It is also responsible for replication and recovery such as replica management with strong consistency, self-healing, and dynamic cluster expansion. It supports a library, called *librados*, to the interface layer so that a client can interact with OSDs.

The storage layer consists of OSDs that actually manage storage devices. Each OSD serves I/O requests from a client and stores/retrieves data in an object unit for which the default size is 4 MB [37]. In addition, OSDs cooperate with each other for replication, migration, and recovery from a failure. The default replication factor is 3. Hence, when data are written by a client, it is divided into objects in *librados*, and then, each object is transferred to 3 OSDs, where the object is actually stored.

There are two representative implementations of OSD, *FileStore* and *BlueStore* [29]. In *FileStore*, an object is stored as a file using kernel-level file systems such as Btrfs and XFS. However, it suffers from poor performance due to the journal of journal anomaly and metadata overhead. This problem led to the development of a new version, *BlueStore*, that has the following two features: (1) accessing raw storage devices directly and (2) making use of RocksDB for storing metadata separately. *BlueStore* is used for this study.

2.2. Mapping Mechanism

The mapping between objects and OSDs in Ceph is governed by CRUSH (Controlled Replication Under Scalable Hashing) [30]. CRUSH is a pseudo-random data distribution algorithm that allows a client to determine OSDs that contain a given object. Note that other distributed storage systems such as GFS [12] and HDFS [13] employ a central node, usually called as master, to manage the mapping between objects and OSDs.

Using the hash-based mapping mechanism instead of the dedicated master node gives several benefits. First, this algorithmic approach can improve performance by removing the need to check the master node to lookup mapping information. Second, it can avoid

a single point of failure, enhancing reliability. Finally, the pseudo-random property of CRUSH can distribute objects uniformly across OSDs, boosting scalability.

Figure 2 illustrates how CRUSH works. Internally, Ceph introduces two concepts, PG (Placement Group) and pool. A PG is defined as a collection of objects, employed for enhancing scalability. Ceph has tens of millions of objects in general, which makes it difficult to manage and track objects individually. To overcome this difficulty, Ceph makes use of PG as a basic management unit for mapping and replication.

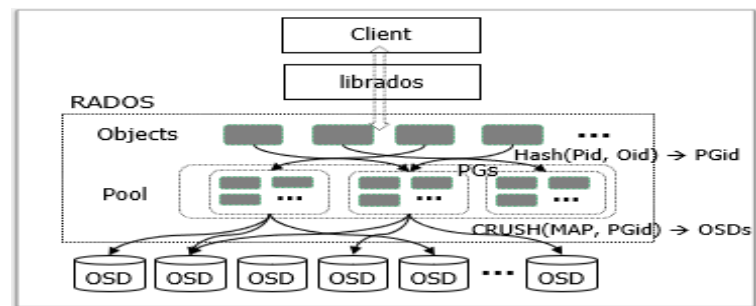


Figure 2. Mapping mechanism in Ceph: an object of a client is associated with a PG (Placement Group), and then a PG is mapped into OSDs using CRUSH (Controlled Replication Under Scalable Hashing).

A pool is defined as a partition of a cluster, which can be created for managing particular types of data or for separating one group of clients from another. When someone creates a pool, he/she can set a number of PGs and a list of OSDs in the pool. A typical configuration uses approximately 100 PGs per OSD to provide optimal balancing [38]. In addition, a pool has CRUSH rules for determining how PGs are mapped into OSDs in the pool. This information is managed by a map, called a CRUSH map. In other words, the map specifies OSDs that belong to a pool and placement rules that dictate how a PG is replicated among OSDs within the pool. By modifying this map, new OSDs can be added or failed OSDs can be removed.

When a client writes an object, RADOS first associates the object with a PG using a simple hash function. In this step, the Pid (Pool ID) and Oid (Object ID) are used as inputs and PGid (Placement Group ID) is calculated as the output of the hash function, as shown in Figure 2. Then, CRUSH uses the CRUSH map and PGid as inputs and determines OSDs to which the object is mapped. By default, an object is mapped into 3 different OSDs for replication. Finally, the object and replicas are transferred to the mapped OSDs and actually stored in storage devices.

2.3. Motivation

Figure 3 shows our first observation. This experiment is based on a Ceph cluster that consists of 5 Intel Core i5-based computers. One computer is used as a client node, while the other 4 computers are used as storage nodes. Two OSDs are created in each storage node, having a total of 8 OSDs in the cluster. Each OSD has one 500 GB 7.2 K RPM WD HDD and one 500 GB Samsung 860EVO SSD. Details of the experimental environment will be further explained in Section 4.

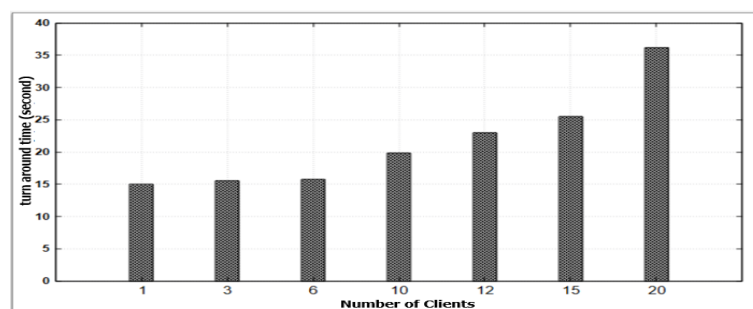


Figure 3. Observation 1: performance degrades as clients increase, which causes the risk of SLA (Service-Level Agreement) violation.

In this cluster, a benchmark program is used by a client. The program is a type of micro-benchmark that writes data for which the size is 20 MB to OSDs and reads it again from OSDs. To estimate the behavior of multiple clients, the programs are executed in parallel, from 1 to 20 as denoted in the x-axis of Figure 3. The y-axis is the average turnaround time of clients.

These results reveal that performance heavily depends on the number of clients due to the resource contention among clients. When a single client runs, the turnaround time is around 15 s. Up until 6 clients, it shows similar performance. However, when the number of clients is beyond 10, performance becomes worse since contention becomes intolerable. This implies that the original Ceph scheme has the risk of SLA violation. For instance, assume that a client requires a turnaround time to be within 20 s. Then, increasing the number of clients cannot meet the requirement. Designing a scheme that can satisfy the requirement even with larger number of clients is the first goal of this study.

Figure 4 presents our second observation. This experiment uses not only HDDs but also SSDs. Figure 4a shows the write latency observed at the storage layer, that is the elapsed time for writing 1 MB data measured at an OSD. It reveals that SSDs outperforms HDDs by more than 3 times. However, the write latency observed at the client layer, which is measured at a client node, does not exhibit such a performance improvement, as shown in Figure 4b.

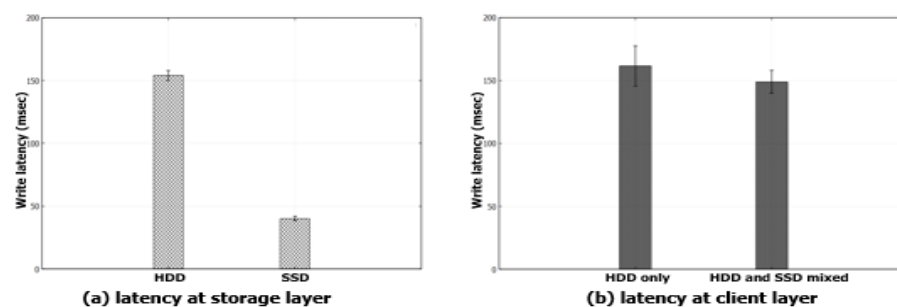


Figure 4. Observation 2: clients do not obtain the benefit of SSDs (Solid State Drives) under the heterogeneity-oblivious approach.

Sensitivity analysis uncovers that the write latency at the client layer is determined by the slowest OSDs. When a client writes an object, it is replicated to multiple OSDs. After all OSDs make the object persistent, RADOS informs of the completion to the client. By the way, the default Ceph configuration is oblivious to storage heterogeneity, not differentiating between HDD-based OSDs and SSD-based OSDs. Hence, many objects are actually replicated on both HDDs and SSDs, and their latency is determined by HDDs. Examining how to make use of these devices in a heterogeneity-aware manner is the second goal in this study.

3. Supporting SLA

This section describes our proposed SLA-aware adaptive mapping scheme and two techniques devised for adaptability. Then, it discusses two use cases on how to make use of heterogeneous storage devices.

3.1. Adaptive Mapping

Guaranteeing SLA needs two mechanisms: SLA specification and enforcement. SLA is a clear definition of the formal agreements about service terms, which can be expressed various ways such as latency, throughput, variation, reliability, and consistency (e.g., at most, 5 min out-of-date [26] or completion within 300 ms [39]). Designing a new SLA specification is out of the scope of this paper. Hence, this paper adopts the existing mechanism and concentrates on how to enforce SLA.

Figure 5 shows the basic idea of our approach. The proposed scheme classifies clients into two groups, namely normal and urgent client. An urgent client is defined as a client who makes a SLA contract. On the contrary, a normal client does not have SLA requirements and simply wishes to use OSDs in a best-effort way at a low price. In addition, the scheme segregates OSDs into two regions, normal and urgent regions. By creating different pools, it can configure that the normal region is used by normal clients while the urgent region is used by urgent ones separately.

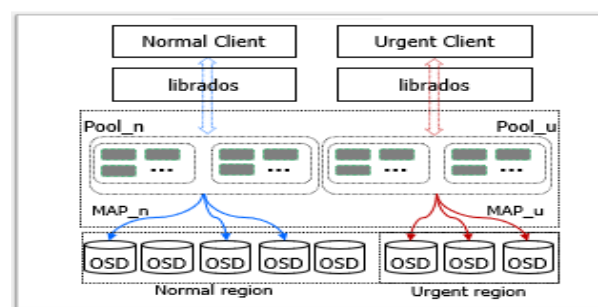


Figure 5. Isolated mode: OSDs are divided into two regions, one for normal clients and the other for urgent clients.

Specifically, for normal clients, it creates a pool, $Pool_n$, with a CRUSH map, Map_n , that consists of OSDs in the normal region. Additionally, for urgent clients, it creates another pool, $Pool_u$, with a CRUSH map, Map_u , consisting of OSDs in the urgent region. As the result, normal and urgent clients are isolated into different regions so that urgent clients are not interfered by normal clients. This mapping is called as an isolated mode.

The size of the urgent region is dictated by SLA requirements of urgent clients. For instance, assume that there are a maximum of 10 urgent clients and at least 30% of OSD bandwidth is required to satisfy an urgent client's SLA. Then, it needs to reserve at least three OSDs for the urgent region. Note that the pseudo-random property of the CRUSH algorithm can distribute urgent clients' requests into OSDs uniformly, which leads to a probabilistic correctness in meeting SLA requirements.

Although this isolated mode can prevent urgent clients from being interfered by normal users, it might lead to low storage utilization. Assume that the number of urgent clients decreases (or becomes zero). Then, maintaining OSDs for urgent clients is actually wasting storage bandwidth and space. In this case, it is better to reallocate them for normal clients. This paper designs a new mode, called shared mode, as shown in Figure 6, where $Pool_n$ uses Map_{all} instead of Map_n , which consists of all OSDs.

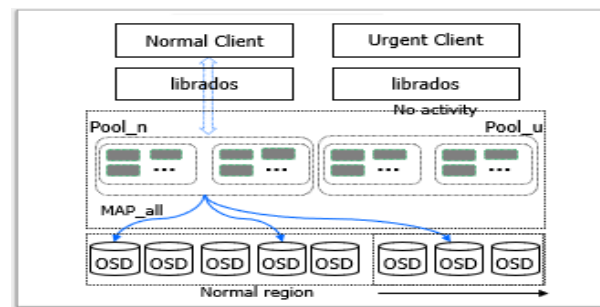


Figure 6. Shared mode: all OSDs are used for normal clients when there is no urgent client.

Our proposed SLA-aware mapping scheme uses these two modes adaptively. It usually operates in isolated mode. When the activity of urgent clients is below a predefined threshold, it changes to the shared mode, thus achieving full performance. When activity increases, it goes back to the isolated mode and supports isolation. The scheme modifies the Ceph monitor daemon that is responsible for keeping track of runtime metrics [38] to measure the activity of urgent clients in real time.

By the way, this adaptability raises two issues. One is how to manage a CRUSH map. In our scheme, an object generated by a normal user can be mapped to either the normal region or all regions adaptively. However, in Ceph, a pool has only one CRUSH map. To tackle this situation, a technique, called *logical cluster*, is devised that abstracts OSDs differently according to the mode. Specifically, in the shared mode, $Pool_n$ manages all OSDs and exports them the normal users, as shown in Figure 6. However, in the isolated mode, it manages OSDs of the normal region only while hiding other OSDs such as Figure 5. This logical different viewpoint of a cluster allows our scheme to switch modes in an efficient way.

The second issue is about data accessibility. When our adaptive scheme switches from the shared mode to the isolated mode, some OSDs cannot be accessed by normal clients, as illustrated in Figure 7. Assume that an object is written by a normal client and that it is stored into an OSD of the urgent region in the shared mode. Then, in the isolated mode, the object cannot be accessed by the client since the OSD is hidden by the logical cluster technique.

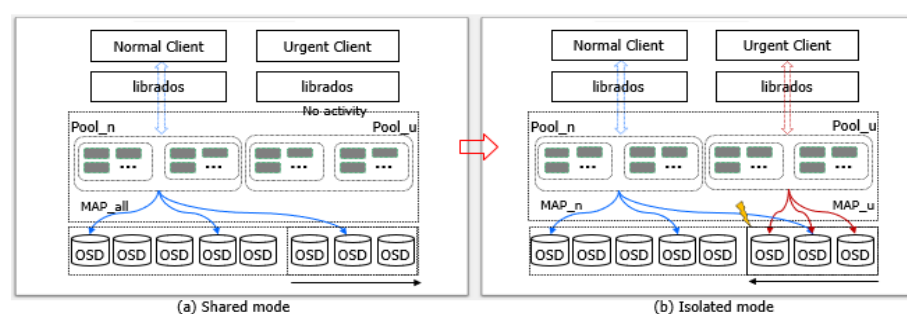


Figure 7. How to orchestrate between the isolated and shared mode.

To overcome this problem, this paper devises a technique, called *normal inclusion*. It makes use of the fact that an object is replicated into multiple OSDs in Ceph for enhancing reliability. The technique imposes that at least one replica of an object is mapped into an OSD in the normal region during the shared mode. Hence, in the isolated mode, the object can be accessed by the normal client since at least one replica is visible to the client.

3.2. Heterogeneous Storage Devices

Employing a new type of storage device is a nontrivial task, especially in distributed storage systems. As already discussed with Figure 4, the introduction of a 3 times faster device does not always lead to a performance improvement of 3 times at the client level.

Various aspects including not only heterogeneity but also management mechanisms and data characteristics need to be considered carefully.

This paper examines two use cases of heterogeneous storage devices, normal/urgent and data/metadata separation, as shown in Figure 8. The normal/urgent separation is the case where relative faster devices (e.g., SSD or SCM (Storage Class Memory) [40]) is used for urgent clients while others are used for normal clients. This use case can provide guaranteed services for urgent clients and best-effort services for normal clients at an affordable price.

The second use case is the data/metadata separation that uses relative faster devices for metadata and others for user data (when we explain this mechanism, we use the term “data” and “user data” interchangeably, which represents data generated by clients). This separation is devices with the consideration of two factors: cost and access pattern. For instance, HDDs are better than SSDs in terms of cost. However, HDDs are notorious for bad performance under the random access pattern due to the slow seek time.

When an OSD manages objects, it also manipulates various metadata such as object identifiers, time, and replica information. Our analysis shows that the access pattern of metadata is random while that of user data is sequential (note that the default size of an object is 4 MB). Moreover, the storage capacity needed for metadata is much smaller than that for user data (6% in our experiment). Therefore, using SSDs for metadata gives an opportunity to enhance performance, while using HDDs for user data makes a positive impact to reduce the total cost of storage.

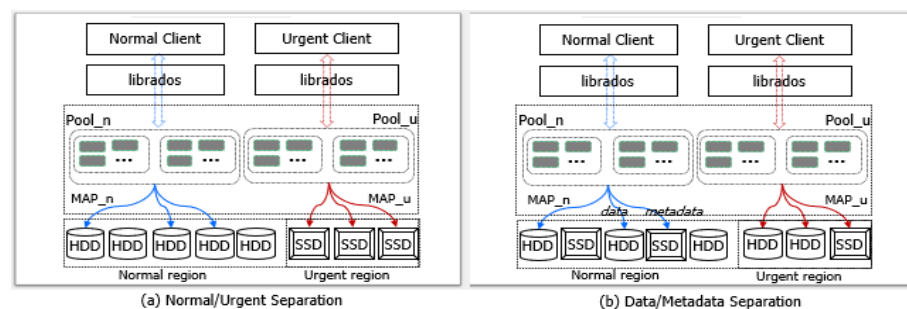


Figure 8. Two use cases for heterogeneous storage devices: separating normal and urgent clients and separating data and metadata.

4. Evaluation

This section explains the experimental methodology. Then, it discusses the evaluation results of adaptive mapping and heterogeneous devices in sequence.

4.1. Experimental Environment

For evaluation, this paper constructs a cluster consisting of five physical computers. Each computer is equipped with 3.30 GHz Intel Core i5 CPU, 24 GB RAM (Intel, Santa Clara, CA, USA), 1Gigabit Ethernet, two 500 GB Samsung 860EVO SDDs (Samsung, Suwon, Korea) and two 500 GB 7.2 K RPM HDD (Dell, Round Rock, TX, USA). On these hardware components, Ubuntu 16.04 server with the Linux kernel 4.15 (Canonical, London, UK) and Ceph Mimic version 13.2.10 (Redhat, Raleigh, NC, USA) are installed. One computer is used for client node and others for storage nodes. The cluster has total 8 OSDs, 2 OSDs per a storage node, where each OSD utilizes HDD only for adaptive mapping, HDD only or SSD only for normal/urgent separation, or both HDD and SSD for the data/metadata separation experiment.

4.2. Effect of Adaptive Mapping

Figure 9 presents the evaluation results of our adaptive mapping scheme. This figure and Figure 3 are results of the same workload, except that the former uses our scheme while the latter uses the Ceph default scheme. In this experiment, an urgent client was

defined as a client who requires a consistent turnaround time that can be obtained when it runs alone in our experimental cluster. Specifically, an urgent client makes an SLA contract that guarantees a turnaround time within 20 s. Clients who do not have such a requirement are treated as normal clients.

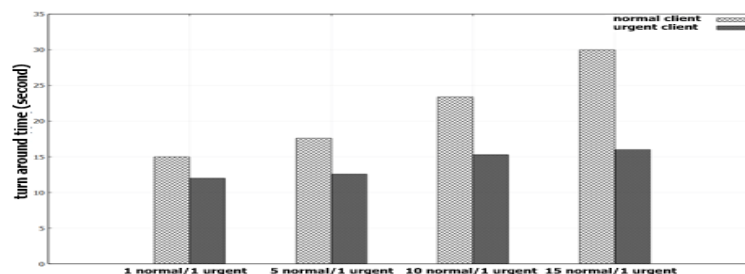


Figure 9. SLA analysis: our proposed adaptive scheme can satisfy the SLA requirement of a urgent client even when the number of normal clients increases.

The results of Figure 9 demonstrate that our proposed scheme indeed guarantees the SLA requirement. The urgent client can obtain a turnaround time below the requirement even though the number of normal clients increases. Note that the original Ceph scheme violates this requirement when the number of clients is larger than 10, as observed in Figure 3, since it does not isolate between urgent and normal clients.

Note that the performance of the urgent client with one normal client is similar to that with 5 normal clients. However, with 10 or more clients, the performance of the urgent client becomes worse. Our analysis uncovers that this degradation is due to the contention of network, not in OSDs. Moreover, experiments with increased numbers of urgent clients show that, for less than 5 urgent clients, our scheme can guarantee the SLA in our experimental cluster.

When comparing Figure 9 with Figure 3, it reveals that our scheme performs worse than the original scheme for normal clients. This degradation is unavoidable since our scheme utilizes less OSDs for normal clients in order to support SLA of the urgent clients. However, if there is no urgent client in the cluster, such degradation is unacceptable. This is why our scheme is designed to operate in an adaptive manner.

Figure 10 shows how the adaptability of our scheme impacts performance. In the figure, the fixed mapping is the scheme that uses the isolated mode only. On the contrary, the adaptive mapping switches between the isolated and shared mode depending whether there is activity from urgent clients. The adaptability indeed enhances performance by up to 18% for 15 normal clients.

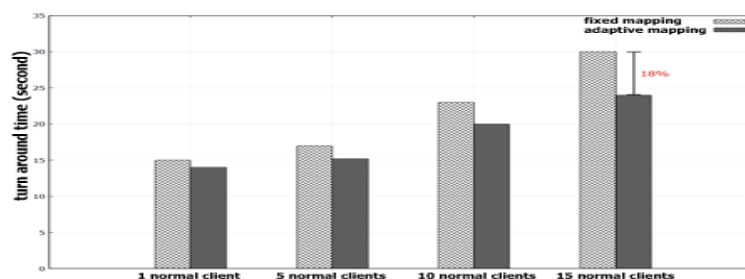


Figure 10. Performance analysis: our adaptive scheme outperforms the fixed scheme when there is no urgent client.

4.3. Effect of Heterogeneous Storage Devices

From now, let us discuss the evaluation results with heterogeneous storage devices.

Figure 11 displays the evaluation results of the normal/urgent separation. Note that, in the heterogeneous-oblivious approach, only replacing some HDDs with SSDs does not enhance performance due to the replication mechanism in Ceph. This is because an object

can be replicated both HDDs and SSDs and its completion time depends on the latency of the slow HDDs, already discussed in Figure 4. On the contrary, our heterogeneous-aware approach can enhance the performance of urgent clients since their objects are replicated among SSDs only. Note that the performance difference between normal and urgent clients is matched well with the difference between HDDs and SSDs at the OSD level.

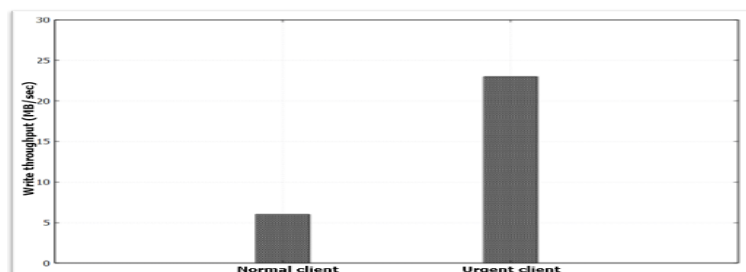


Figure 11. Effect of the normal/urgent separation: the heterogeneity-aware approach can obtain the benefit of SSDs at client layer (compare with Figure 4).

Figure 12 presents the evaluation results of data/metadata separation under four different configurations. In the HDD configuration, each OSD stores both user data and metadata in an HDD. Similarly, all data are stored in an SSD in the SSD configuration. In the HDD+SSD configuration, each OSD utilizes two devices, one HDD for user data and one SSD for metadata. Finally, in the SSD+SSD configuration, each OSD utilizes two SSDs, one for user data and the other for metadata.

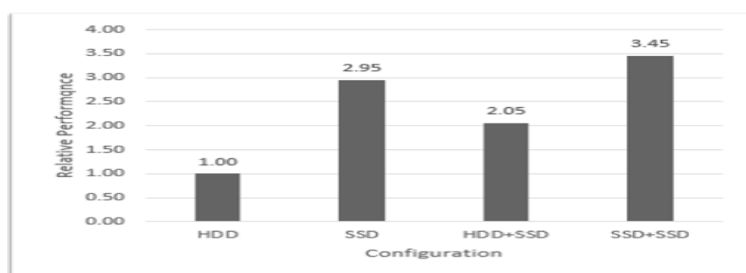


Figure 12. Effect of the normal/urgent separation: fast device, data-awareness, and parallelism enhance performance.

The results show that making an SSD-only cluster can enhance performance by up to 2.95 times, compared with a HDD-only cluster. Separating data and metadata into HDD and SSD can enhance performance around 2 times by reducing the metadata overhead. Utilizing two SSDs can enhance it further by exploiting parallelism between data and metadata processing.

The performance per cost ratio under the four different configurations are given in Figure 13. There are three graphs in the figure with different cost ratios between HDDs and SSDs, namely 1 to 2, 1 to 5, and 1 to 10, respectively. When the cost ratio between HDDs and SSDs is small, an SSD-only cluster can be a feasible solution. However, when the ratio becomes larger, the data/metadata separation is the best solution in terms of the performance per cost.

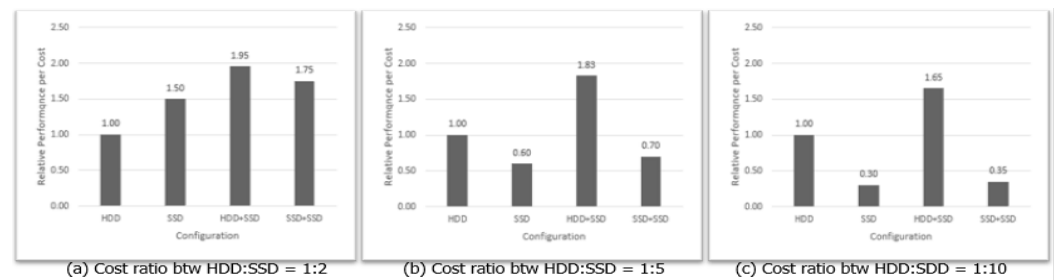


Figure 13. Performance over cost ratio: the normal/urgent separation is a cost-effect solution.

5. Related Work

SLA management is a topic that has been studied actively in various research area, especially in cloud and storage research areas.

Alhamad et al. discussed several issues that need to be considered to design SLA in cloud computing [27]. They also investigated the negotiation strategies and methods to maintain trust between a cloud provider and user. DeCandia et al. designed Dynamo, a distributed key-value store for cloud, where SLAs are expressed as the response-time of put/get operations and the 99.9th percentile of the total number of requests [39]. Ortiz et al. proposed a scheme, called SLAOrchestrator, that guarantees per-query execution time in cloud analytics [41]. Gulati et al. devised mClock, an algorithm that supports proportional-share fairness of IO allocation in virtualization environments [42].

Storage researchers also explore how to comply with SLA at the storage level. For achieving SLA, Tanimura et al. suggested a distributed storage system that makes a time-based I/O access and storage space reservation [25]. Terry et al. examined the tradeoff among consistency, availability, and performance in distributed storage [26]. Then, they designed a system, where a client can declare his/her consistency requirement (e.g., strong, eventual, or read-my-writes) and a provider adjusts the client's latency to match the required consistency.

Shue et al. presented Pisces, a scheme that supports per-tenant fairness in a shared storage system [43]. It was based on four techniques: partition placement, weight allocation, replica selection, and weighted fair queuing. Ardekani et al. proposed a scheme that reconfigures the number of replicas and their locations so that it can enhance performance while guaranteeing SLA [44].

Ceph characteristics are important when a SLA scheme is designed in Ceph. BlueStore is the recent implementation of Ceph that makes use of RocksDB for metadata management [29]. Lee et al. evaluated three different Ceph OSD implementations, namely FileStore, KStore, and BlueStore, and analyzed their write behaviors, especially focusing on WAF (Write Amplification Factor) [37]. Wang et al. investigated the scalability of Ceph for scientific high-performance computing environments [45]. Chagam et al. presented basic configuration guidelines for Ceph deployment [46].

Two studies are closely related to our work. Wang et al. observed that CRUSH-based storage systems suffer from uncontrolled data migration when expanding cluster [47]. To overcome this problem, they extended CRUSH mapping to differentiate new and old OSDs and to supervise migration OSDs. Their work was similar to ours in that both make use of mapping in Ceph. However, their goal was enhancing performance during cluster expansion while ours is guaranteeing SLA in Ceph.

Wu and Brandt designed a new framework, called Bourbon, to provide different services into different classes of workloads in Ceph [48]. Their objective was the same as ours. However, their proposal differs from ours in that it exploits a new I/O queuing and cache mechanism. In other words, their approach is a scheduling-based scheme while ours is a mapping-based and heterogeneous-aware approach.

6. Conclusions

The contribution of this paper can be summarized in two aspects. As a scientific contribution, this paper introduces a novel adaptive mapping scheme that provides SLA guarantee for urgent clients by preventing them from being interfered by normal clients. In addition, it devises two techniques, logical cluster and normal inclusion, to realize the adaptability in an efficient way.

As an academic contribution, this paper raises several issues such as client-type-based separation and data-type-based separation when heterogeneous storage devices are employed in distributed storage systems. Additionally, it presents various implementation-based evaluation results and experimental methodologies, which will be helpful for researchers to study distributed storage systems.

There are two limitations in this study. The first one is that the proposed logical cluster relies on the Ceph CRUSH algorithm that was originally designed without considering heterogeneous storage devices. Hence, to achieve optimal performance, it needs to address not only the mapping mechanism but also the CRUSH algorithm itself so that it operates with different replica management policies according to the characteristics of heterogeneous storage devices. The second limitation is that, even though the normal inclusion allows for data accessibility, the number of replicas in an isolated mode can be reduced, with the potential to weaken data availability.

There are three research directions as future work. First, our proposal will be evaluated more quantitatively under diverse workloads and Ceph configuration parameters. The second direction is designing new replica management policies for heterogeneous storage devices. Our third research direction is extending BlueStore based on new types of SSDs such as Optane SSDs [49] and ZNS SSDs [50].

Author Contributions: Conceptualization, J.C. and H.P.; Methodology, S.C.; Software, S.C.; Validation, S.C.; Writing—original draft preparation, J.C. and H.P.; Writing—review and editing, J.C.; Supervision, J.C.; Project administration, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2019R1F1A1062284).

Acknowledgments: The authors appreciate all the reviewers and editors for their precious comments and work on this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Akter, S.; Wamba, F. Big Data Analytics in E-commerce: A Systematic Review and Agenda for Future Research. Faculty of Business—Papers (Archive), 886. Available online: <https://ro.uow.edu.au/buspapers/886> (accessed on 28 January 2021).
2. Formica, A.; Pourabbas, E.; Taglino, F. Semantic Search Enhanced with Rating Scores. *Future Internet* **2020**, *12*, 67. [CrossRef]
3. Blazquez, D.; Domenech, J. Big Data sources and methods for social and economic analyses. *Technol. Forecast. Soc. Chang.* **2018**, *130*, 99–113. [CrossRef]
4. Najafabadi, M.N.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*, 1. [CrossRef]
5. Serrano, W. Neural Networks in Big Data and Web Search. *Data* **2019**, *4*, 7. [CrossRef]
6. Qi, G.; Luo, J. Small Data Challenges in Big Data Era: A Survey of Recent Progress on Unsupervised and Semi-Supervised Methods. Available online: <https://arxiv.org/abs/1903.11260> (accessed on 28 January 2021).
7. Amini, S.; Gerostathopoulos, I.; Prehofer, C. Big data analytics architecture for real-time traffic control. In Proceedings of the 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Naples, Italy, 26–28 June 2017.
8. Xu, F.; Zheng, H.; Jiang, H.; Shao, W.; Liu, H.; Zhou, Z. Cost-Effective Cloud Server Provisioning for Predictable Performance of Big Data Analytics. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1036–1051. [CrossRef]
9. Trivedi, A.; Stuedi, P.; Pfefferle, J.; Schuepbach, A.; Metzler, B. Albis: High-Performance File Format for Big Data Systems. In Proceedings of the USENIX Annual Technical Conference (ATC), Boston, MA, USA, 11–13 July 2018.
10. Orenga-Roglá, S.; Chalmers, R. Framework for Implementing a Big Data Ecosystem in Organizations. *Commun. ACM* **2019**, *62*, 58–65. [CrossRef]

11. Patrizio, A. IDC: Expect 175 Zettabytes of Data Worldwide by 2025. Available online: <https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html> (accessed on 28 January 2021).
12. Ghemawat, S.; Gobioff, H.; Leung, S.T. The Google File System. In Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), Bolton Landing (Lake George), New York, NY, USA, 19–22 October 2003.
13. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In Proceedings of the 26th IEEE Symposium on Massive Storage Systems and Technologies (MSST), Lake Tahoe, NV, USA, 3–7 May 2010.
14. Weil, S.A.; Brandt, S.A.; Miller, E.L.; Long, D.D.E.; Maltzahn, C. Ceph: A Scalable, High-Performance Distributed File System. In Proceedings of the 7th symposium on Operating Systems Design and Implementation (OSDI), Seattle, WA, USA, 6–8 November 2006.
15. Huang, C.; Simitci, H.; Xu, Y.; Ogun, A.; Calder, B.; Gopalan, P.; Li, J.; Yekhanin, S. Erasure Coding in Windows Azure Storage. In Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST), San Jose, CA, USA, 14–17 February 2012.
16. Palankar, M.; Iamnitchi, A.; Ripeanu, M.; Garfinkel, S. Amazon S3 for Science Grids: A Viable Solution? In Proceedings of the International Workshop on Data-Aware Distributed Computing (DADC), Boston, MA, USA, 24–27 June 2008.
17. Kapadia, A.; Varma, S.; Rajana, K. *Implementing Cloud Storage with OpenStack Swift*, 1st ed.; Packt Publishing: Birmingham, UK, 2014.
18. Beaver, D.; Kumar, S.; Li, H.C.; Sobel, J.; Vajgel, P. Finding a needle in Haystack: Facebook’s photo storage. In Proceedings of the USENIX Annual Technical Conference (ATC), Boston, MA, USA, 22–25 June 2006.
19. Lustre Architecture. Available online: <http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf> (accessed on 28 January 2021).
20. GlusterFS Architecture. Available online: <https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/> (accessed on 28 January 2021).
21. Zhou, W.; Wang, W.; Hua, X.; Zhang, Y. Real-Time Traffic Flow Forecasting via a Novel Method Combining Periodic-Trend Decomposition. *Sustainability* **2020**, *12*, 5891. [CrossRef]
22. Chen, G.; Wiener, J.L.; Iyer, S.; Jaiswal, A.; Lei, R.; Simha, N.; Wang, W.; Wilfong, K.; Williamson, T.; Yilma, S. Realtime Data Processing at Facebook. In Proceedings of the 2016 International Conference on Management of Data (SIGMOD), San Francisco, CA, USA, 26–30 June 2016.
23. Divakaran, D.M.; Le, T.; Gurusamy, M. An Online Integrated Resource Allocator for Guaranteed Performance in Data Centers. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 6. [CrossRef]
24. Chuang, J.C.; Sirbu, M. Distributed network storage service with quality-of-service guarantees. *J. Netw. Comput. Appl.* **2000**, *23*, 163–185. [CrossRef]
25. Tanimura, Y.; Hidetaka, K.; Kudoh, T.; Kojima, I.; Tanaka, Y. A distributed storage system allowing application users to reserve I/O performance in advance for achieving SLA. In Proceedings of the 11th ACM/IEEE International Conference on Grid Computing (GRID), Brussels, Belgium, 25–28 October 2010.
26. Terry, D.B.; Prabhakaran, V.; Kotla, R.; Balakrishnan, M.; Aguilera, M.K.; Abu-Libdeh, H. Consistency-Based Service Level Agreements for Cloud Storage. In Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP), Farmington, PA, USA, 3–6 November 2013.
27. Alhamad, M.; Dillon, T.; Chang, E. Conceptual SLA framework for cloud computing. In Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies, Dubai, United Arab Emirates, 14–16 April 2010.
28. Chum, S.; Li, J.; Park, H.; Choi, J. SLA-Aware Adaptive Mapping Scheme in Bigdata Distributed Storage Systems. In Proceedings of the 9th International Conference on Smart Media and Applications (SMA), Jeju, Korea, 17–19 September 2020.
29. Aghayev, A.; Weil, S.A.; Kuchnik, M.; Nelson, M.; Ganger, G.R.; Amvrosiadis, G. File Systems Unfit as Distributed Storage Backends: Lessons from 10 Years of Ceph Evolution. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP), Huntsville, ON, Canada, 27–30 September 2019.
30. Weil, S.A.; Brandt, S.A.; Miller, E.L.; Maltzahn, C. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing, Tampa, FL, USA, 11–17 November 2006.
31. D’Atri, A.; Bhembre, V.; Singh, K. *Learning Ceph: Unified, Scalable, and Reliable Open Source Storage Solution*, 2nd ed.; Packt Publishing: Birmingham, UK, 2017.
32. Ceph Storage Datasheet. Available online: <https://www.redhat.com/en/resources/ceph-storage-datasheet> (accessed on 28 January 2021).
33. Mellanox White Paper. Installing Hadoop over Ceph, Using High Performance Networking. Available online: https://www.mellanox.com/related-docs/whitepapers/wp_hadoop_on_cephfs.pdf (accessed on 28 January 2021).
34. Chien, S.; Markidis, S.; Sishtla, C.P.; Santos, L.; Herman, P.; Narasimhamurthy, S.; Laure, E. Characterizing Deep-Learning I/O Workloads in TensorFlow. In Proceedings of the IEEE/ACM 3rd International Workshop on Parallel Data Storage and Data Intensive Scalable Computing Systems (PDSW-DISCS), Dallas, TX, USA, 12 November 2018.
35. Yang, C.; Liu, J.; Kristiani, E.; Liu, M.; You, I.; Pau, G. NetFlow Monitoring and Cyberattack Detection Using Deep Learning with Ceph. *IEEE Access* **2020**, *8*, 7842–7850. [CrossRef]
36. Weil, S.A.; Leung, A.W.; Brandt, S.A.; Maltzahn, C. RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters. In Proceedings of the 2nd Parallel Data Storage Workshop (PDSW), Reno, NV, USA, 11 November 2007.
37. Lee, D.; Jeong, K.; Han, S.; Kim, J.; Hwang, J.; Cho, S. Understanding Write Behaviors of Storage Backends in Ceph Object Store. In Proceedings of the 33rd International Conference on Massive Storage Systems and Technology (MSST), Santa Clara, CA, USA, 15–19 May 2017.

38. Introduction to Ceph. Available online: <https://docs.ceph.com/en/latest/start/intro/> (accessed on 28 January 2021).
39. De Candia, G.; Hastorun, D.; Jampani, M.; Kakulapati, G.; Lakshman, A.; Pilchin, A.; Sivasubramanian, S.; Voshall, P.; Vogels, W. Dynamo: Amazon's Highly Available Key-value Store. In Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP), Stevenson, WA, USA, 14–17 October 2007.
40. Wu, X.; Qiu, S.; Reddy, N.L. SCMFS: A File System for Storage Class Memory and its Extensions. *ACM Trans. Storage* **2013**, *9*, 1822–1833. [[CrossRef](#)]
41. Ortiz, J.; Lee, B.; Balazinska, M.; Gehrke, J.; Hellerstein, J.L. SLAOrchestrator: Reducing the Cost of Performance SLAs for Cloud Data Analytics. In Proceedings of the USENIX Annual Technical Conference (ATC), Boston, MA, USA, 11–13 July 2018.
42. Gulati, A.; Merchant, A.; Varman, P.J. mClock: Handling Throughput Variability for Hypervisor IO Scheduling. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Vancouver, BC, Canada, 4–6 October 2010.
43. Shue, D.; Freedman, M.J.; Shaikh, A. Performance Isolation and Fairness for Multi-Tenant Cloud Storage. In Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Hollywood, CA, USA, 8–10 October 2012.
44. Ardekani, M.S.; Terry, D.B. A Self-Configurable Geo-Replicated Cloud Storage System. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Bloomfield, CO, USA, 6–8 October 2014.
45. Wang, F.; Nelson, M.; Oral, S.; Atchley, S.; Weil, S.A.; Settlemeyer, B.W.; Caldwell, B.; Hill, J. Performance and Scalability Evaluation of the Ceph Parallel File System. In Proceedings of the 8th Parallel Data Storage Workshop (PDSW), Denver, CO, USA, 18 November 2013.
46. Chagam, A.; Ferber, D.; Leone, D.J.; Moreno, O.; Wang, Y.; Zhang, Y.; Zhang, J.; Zou, Y.; Henderson, M.W. Intel Solutions for Ceph Deployments. Available online: https://builders.intel.com/docs/storagebuilders/Intel_solutions_for_ceph_deployments.pdf (accessed on 28 January 2021).
47. Wang, L.; Chuxing, D.; Zhang, Y.; Xu, J.; Xue, G. MAPX: Controlled Data Migration in the Expansion of Decentralized Object-Based Storage Systems. In Proceedings of the 18th USENIX Conference on File and Storage Technologies (FAST), Santa Clara, CA, USA, 24–27 February 2020.
48. Wu, J.C.; Brandt, S.A. Providing Quality of Service Support in Object-Based File System. In Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies (MSST), San Diego, CA, USA, 24–27 September 2007.
49. Wu, K.; Arpaci-Dusseau, A.; Arpaci-Dusseau, R. Towards an Unwritten Contract of Intel Optane SSD. In Proceedings of the 11th USENIX Workshop on Hot Topics in Storage and File Systems, Renton, WA, USA, 8–9 July 2019.
50. Choi, G.; Oh, M.; Lee, K.; Choi, J.; Jin, J.; Oh, Y. A New LSM-style Garbage Collection Scheme for ZNS SSDs. In Proceedings of the 12th USENIX Workshop on Hot Topics in Storage and File Systems, Virtual event, 13–14 July 2020.