A Study on the Characteristics of Ceph KStore

Sanghoon Han, Kisik Jeong, Dong-Yun Lee, and Jin-Soo Kim
Sungkyunkwan University, Suwon, Korea

1. Introduction

Ceph is one of the most widely used open-source distributed object store and file system designed to provide high reliability, scalability, and performance [1]. Internally, Ceph uses three types of storage backends: FileStore, KStore, and BlueStore. Among them, KStore is very simple which stores all the data and metadata in a key-value store such as LevelDB and RocksDB. Also, several attempts are being made to use Ethernet-enabled, key-value disk drives such as Kinetic Open Storage [2] as a storage backend for Ceph. However, tuning the underlying key-value store to maximize the performance of Ceph KStore requires considerable efforts including the deployment of a real Ceph testbed, configuring the key-value store to work with Ceph, and the repeated tests varying one or more parameters.

This paper proposes an alternative approach where the underlying key-value stores can be optimized according to the characteristics of Ceph KStore without running them on Ceph. For this, we have slightly modified Ceph to record the trace of key-value operations issued from Ceph KStore while running a given workload. One can use these traces to extract valuable information on the characteristics of Ceph KStore, such as key length, value length, etc. Also, these traces can be replayed on a particular key-value store to estimate its performance under Ceph KStore.

2. Analysis of Key-Value Patterns

As an example of our approach, we first show our analysis results on the key-value patterns obtained while a client performs 4KB random writes using `fio` benchmark over a 4GB block storage on Ceph.

We find Ceph KStore uses three types of key-value operations, namely `get`, `set`, and `rmkey`. The `set` and `rmkey` operations are used to change the contents of database and they are bundled together in a transaction. Each transaction corresponds to a single write request of a client.

In the Ceph Hammer version (v0.94.9), a transaction consists of 10 operations, each of which uses a different key prefix such as "`STRIP`", "`OBJOMAP`", and "`OBJATTR`." In particular, the key with "`STRIP`" means the actual data, whose value is 4KB in size. The key-value operations originating from write transactions are responsible for 93.6% of the total number of key-value operations.

The trend in the Ceph Jewel version (v10.2.3) is similar to that in the Hammer version. However, the notable difference is that the actual data is stored in a unit of 64KB instead of 4KB. Also, the number of operations in a transaction has been reduced to four or five operations. In addition, we find that the key prefix has been changed intoto a more compact form such as "`D`", "`M`", and "`O`" instead of "`STRIP`", "`OBJOMAP`", and "`OBJATTR`.".

3. Analysis of Write Amplification

Now we present that replaying the trace gives the almost identical results with our evaluations on WAF (Write Amplification Factor) with two key-value stores (LevelDB and RocksDB) on two Ceph versions (Hammer and Jewel). WAF represents the ratio of the amount of data written into the storage to the amount of actual data written by the client. Since WAF is increased due to additional writes in the Ceph layer, WAF can be used to evaluate the efficiency of the Ceph storage backend.

We use `ftrace` to analyze the WAF of Ceph KStore. We distinguish metadata and journaling writes of the file system by metadata flag and the logical block number. Then, we assume that the rest of the writes are generated by the key-value store. The amount of writes incurred during compaction in LevelDB and RocksDB has been identified using the log files generated by each key-value store. The trace replayer is developed by cloning the KeyValueDB, which is a translation layer used by Ceph to communicate with the backend key-value store.

Our evaluation indicates that LevelDB gives lower WAF values compared to RocksDB in both Ceph versions. In both LevelDB and RocksDB, the WAF value is increased by 7~8x when we move from Hammer to Jewel. This is due to the larger write unit size (64KB) in Jewel. Finally, the WAF values obtained from the replayer have been deviated only by 1.5~4.5% from the actual value.

5. References

[1] Ceph, http://ceph.com/, 2017.

[2] Kinetic Open Storage Project, http://openkinetic.org, 2017.

A Study on the Characteristics of Ceph KStore

Sanghoon Han, Kisik Jeong, Dong-Yun Lee, and Jin-Soo Kim

Sungkyunkwan University, Korea

Motivations

Optimizing Ceph KStore Requires

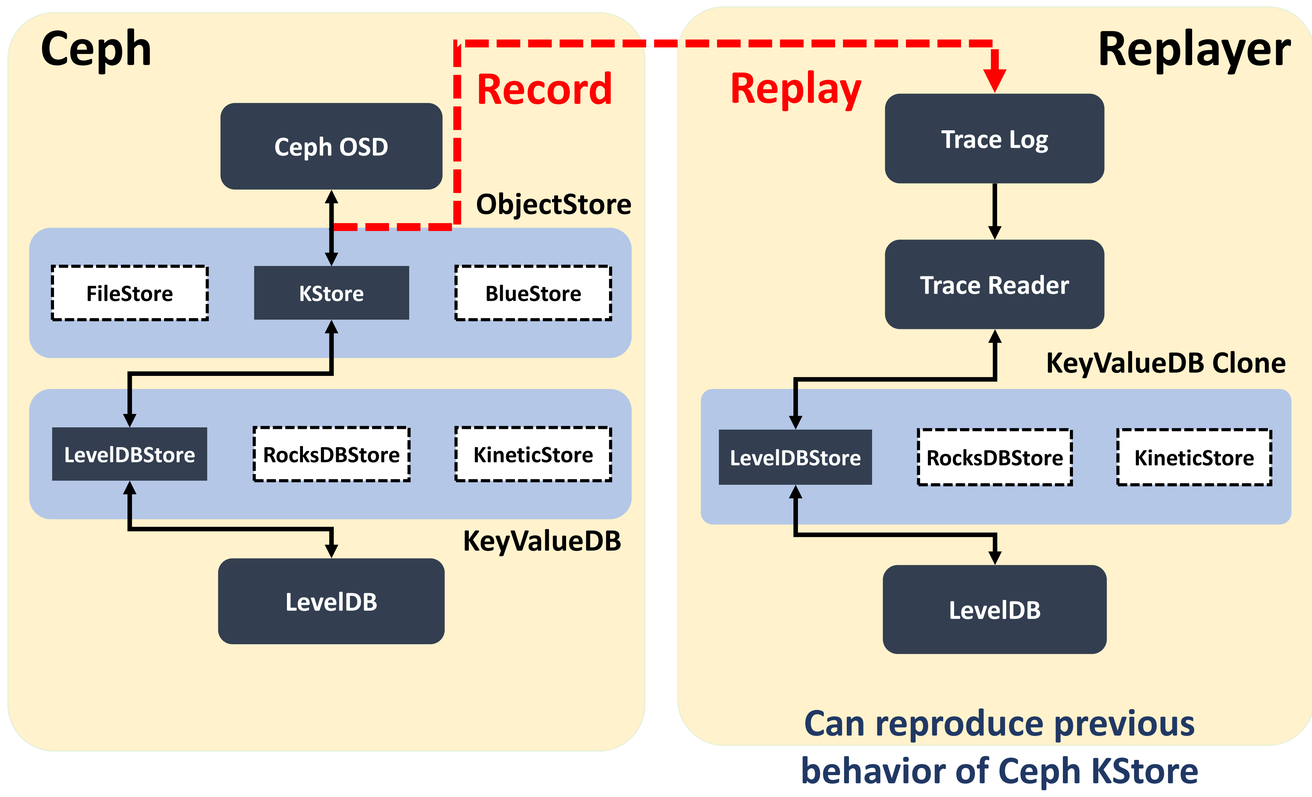
- The deployment of a real Ceph testbed
- Configuring the key-value store to work with Ceph
- Repeated experiments with various parameters settings

It is difficult to perform quickly and easily

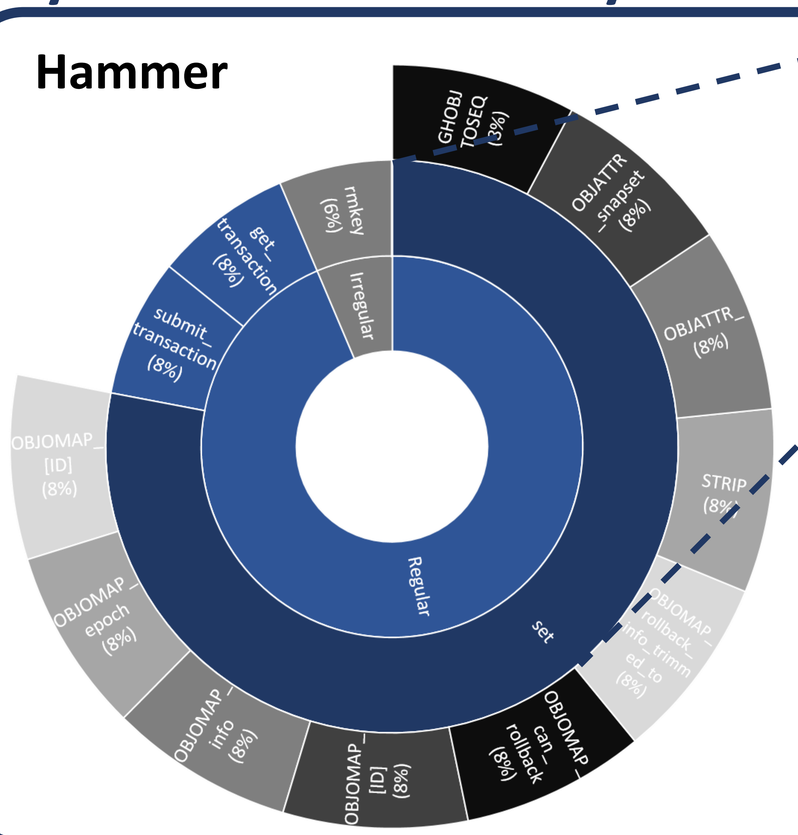
Our Approach

- Modifying Ceph to record the trace of key-value operations issued from Ceph Kstore
- Analyzing key-value pattern of recorded traces
- Implementing Replayer to run the recorded traces on a particular key-value store
- Comparing WAF of Replayer with actual Ceph

Design Overview



Key-Value Pattern Analysis



#	Prefix	Key Length min	Key Length max	Key Length avg	Value Length min	Value Length max	Value Length avg
1	OBJOMAP_[ID]	64	64	64	175	175	175
2	OBJOMAP_epoch	39	39	39	4	4	4
3	OBJOMAP_info	38	38	38	730	730	730
4	OBJOMAP_[ID]	64	64	64	175	175	175
5	OBJOMAP_can_rollback	48	48	48	12	12	12
6	OBJOMAP_rollback_info_trimmed_to	57	57	57	12	12	12
7	STRIP	30	33	31.916	4096	4096	4096
8	OBJATTR__	34	34	34	250	250	250
9	OBJATTR_snapset	40	40	40	31	31	31
10	GHOBJTOSEQ	77	78	77.737	155	1178	1171.665

Regular, Set : 78%

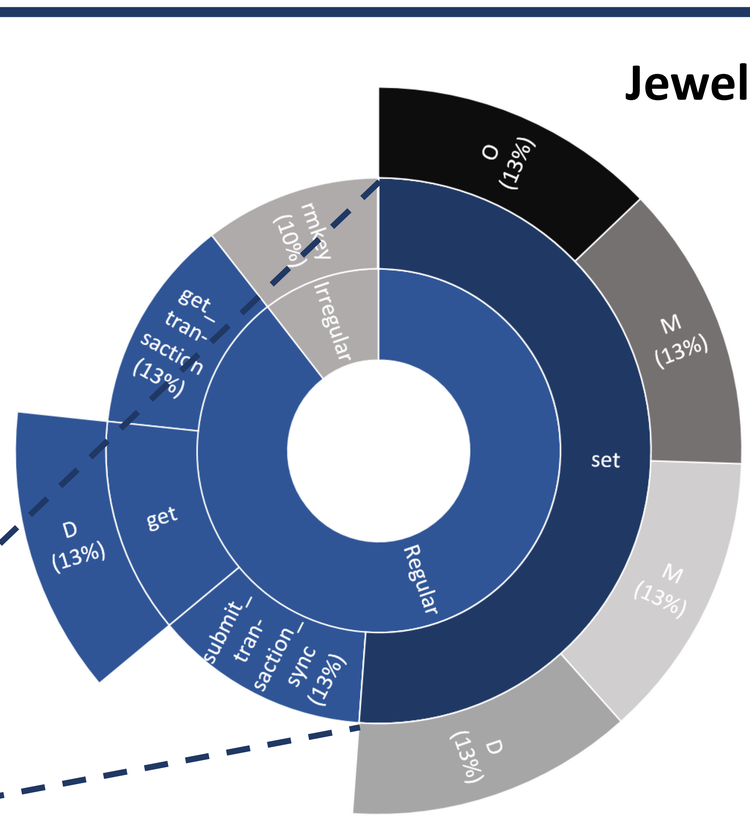
4KB/Key

User Data

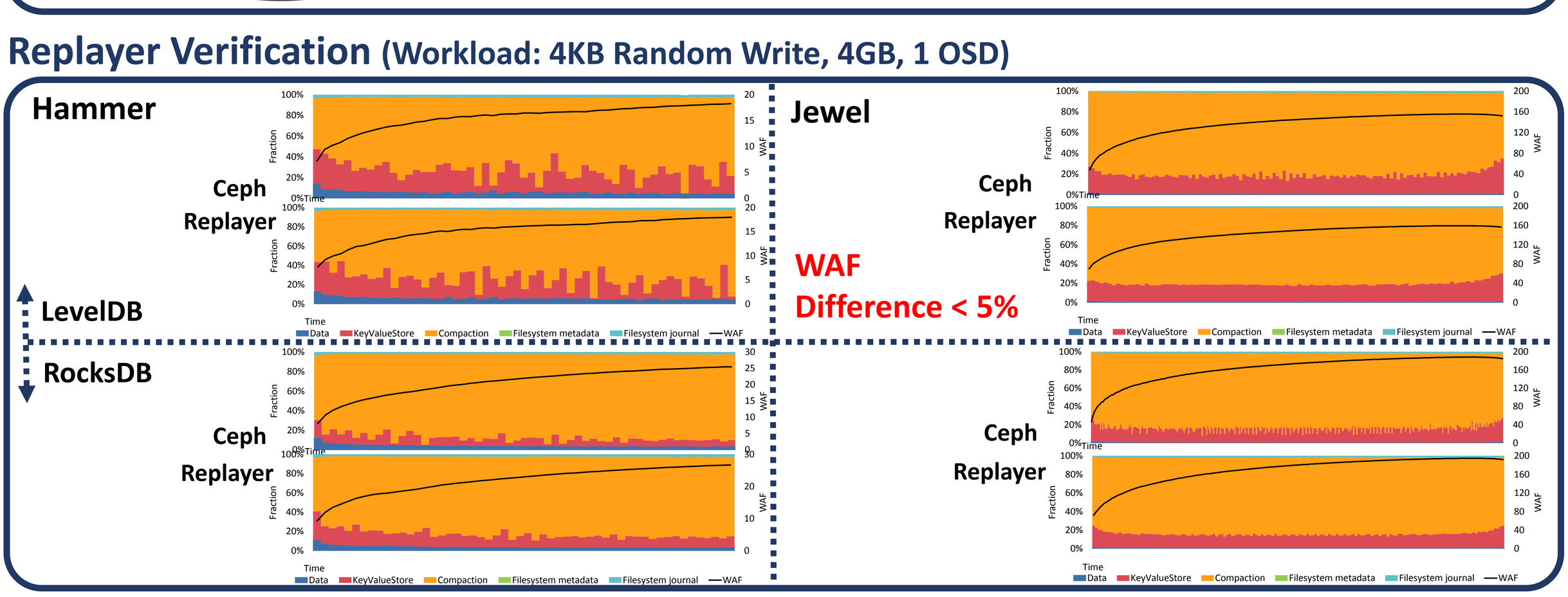
64KB/Key

Regular, Set : 52%

#	Prefix	Key Length Min	Key Length Max	Key Length Average	Value Length Min	Value Length Max	Value Length Average
1	D	18	18	18	4096	65536	34816.1
2	M	42	42	42	182	182	182
3	M	16	16	16	847	847	847
4	O	74	74	74	374	374	374



Replayer Verification (Workload: 4KB Random Write, 4GB, 1 OSD)



Hammer: v0.94.9

Jewel: v10.2.3