Bluestore & rocksdb optimization

Li, Xiaoyan

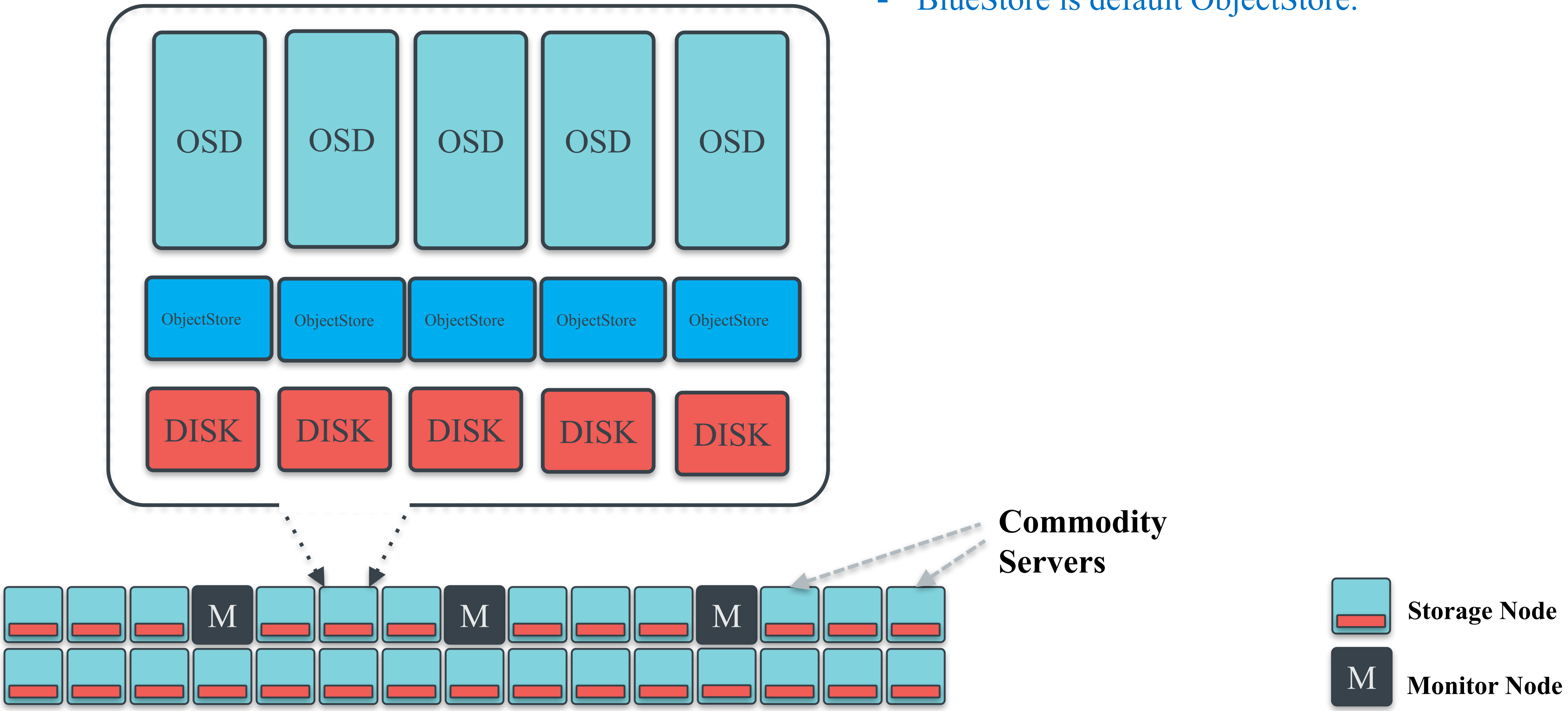
Agenda

- BlueStore overview
- Rocksdb overview
- BlueStore latency over OSD
- Rocksdb optimization

Agenda

- **BlueStore overview**
- Rocksdb overview
- BlueStore latency over OSD
- Rocksdb optimization

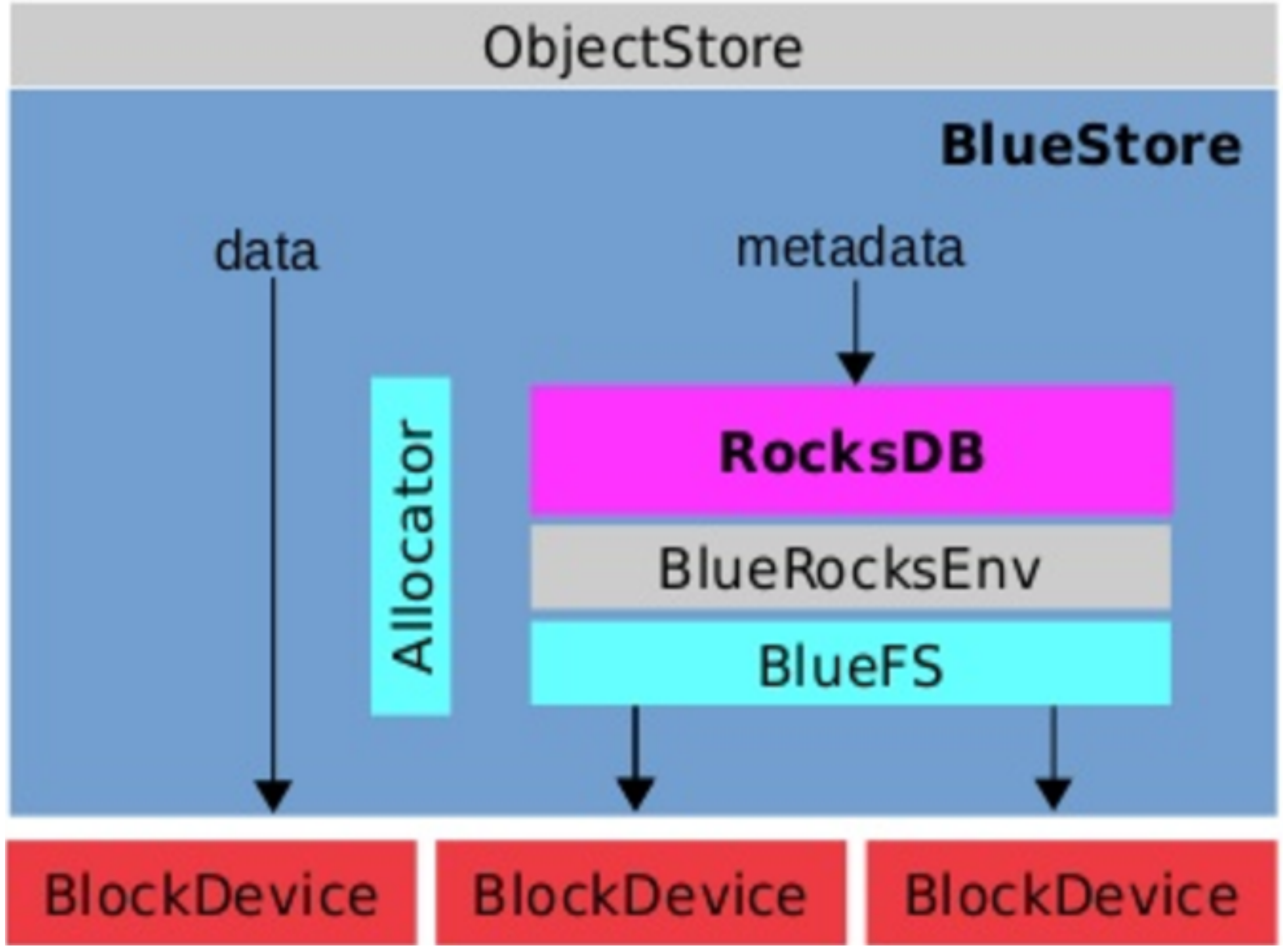
Ceph: Architecture



- BlueStore is default ObjectStore.

BlueStore

- BlueStore = Block + NewStore
  - ➢ Data written directly to block device
  - ➢ Key/value database (Rocksdb) for metadata
  - ➢ Light weight file system BlueFS.



Metadata

S* - “superblock” properties for the entire store

B* - block allocation metadata (blocks, size, blocks_per_key etc)

b* - allocation bitmap

T* - stats (bytes used, compressed, tec)

C* - collection name -> cnode_t

O* - object name -> onode mapping

X* - shared blobs

L* - deferred writes

M* - omap data

Metadata – cons.

- What kind of metadata?

4k random write:

Total 11120873 key-value pairs

L,3177321

M,4764738

O,3178352

size of keys (MB):

L,30

M,157

O,228

size of values (MB):

L,6264

M,578

O,2202

total 9044

16k random write:

Total 11113220 key-value pairs

M,4758712

O,3182396

b,3172112

size of keys (MB):

M,157

O,229

b,30

size of values (MB):

M,577

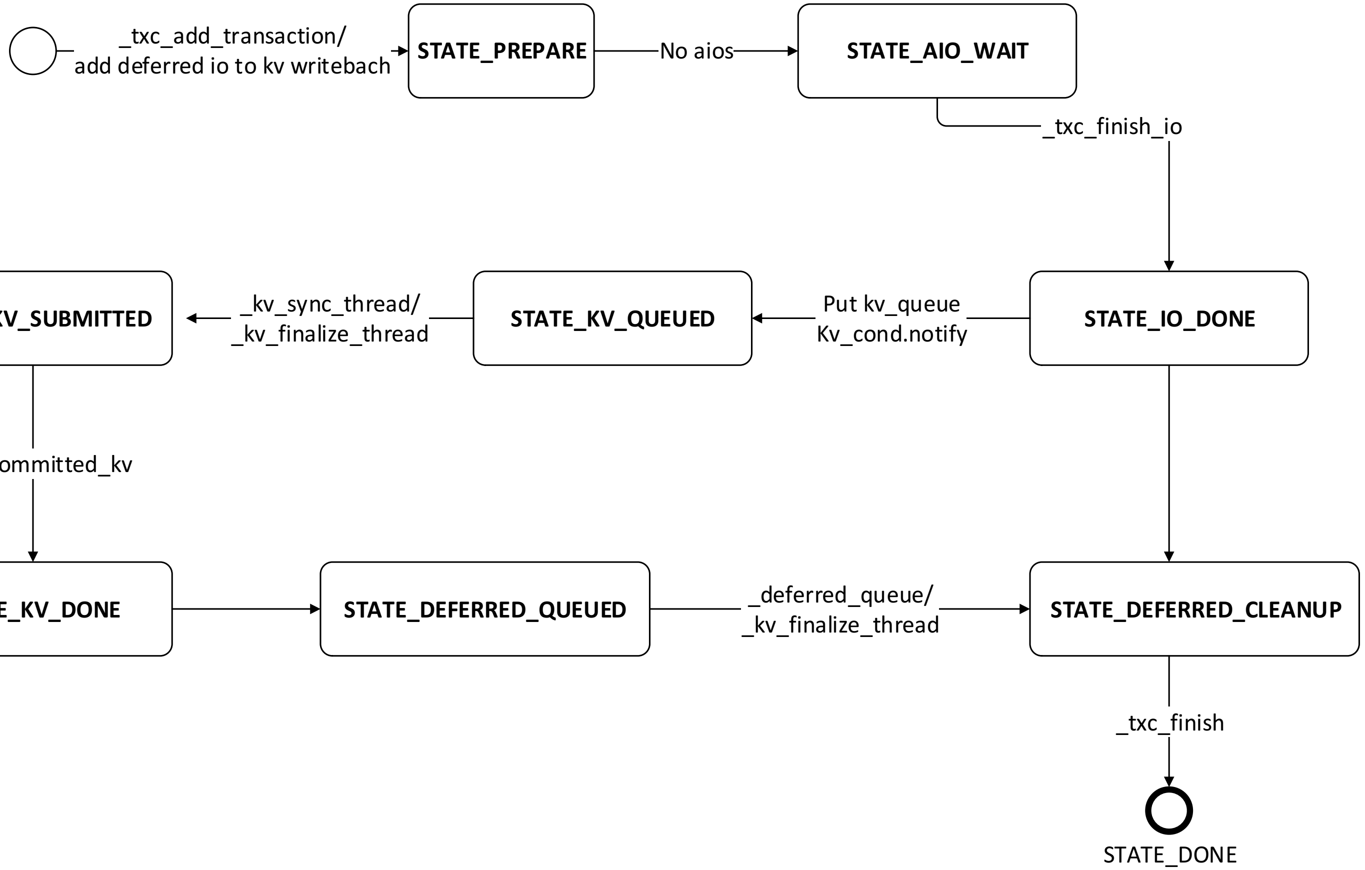
O,1580

b,48

total 2205

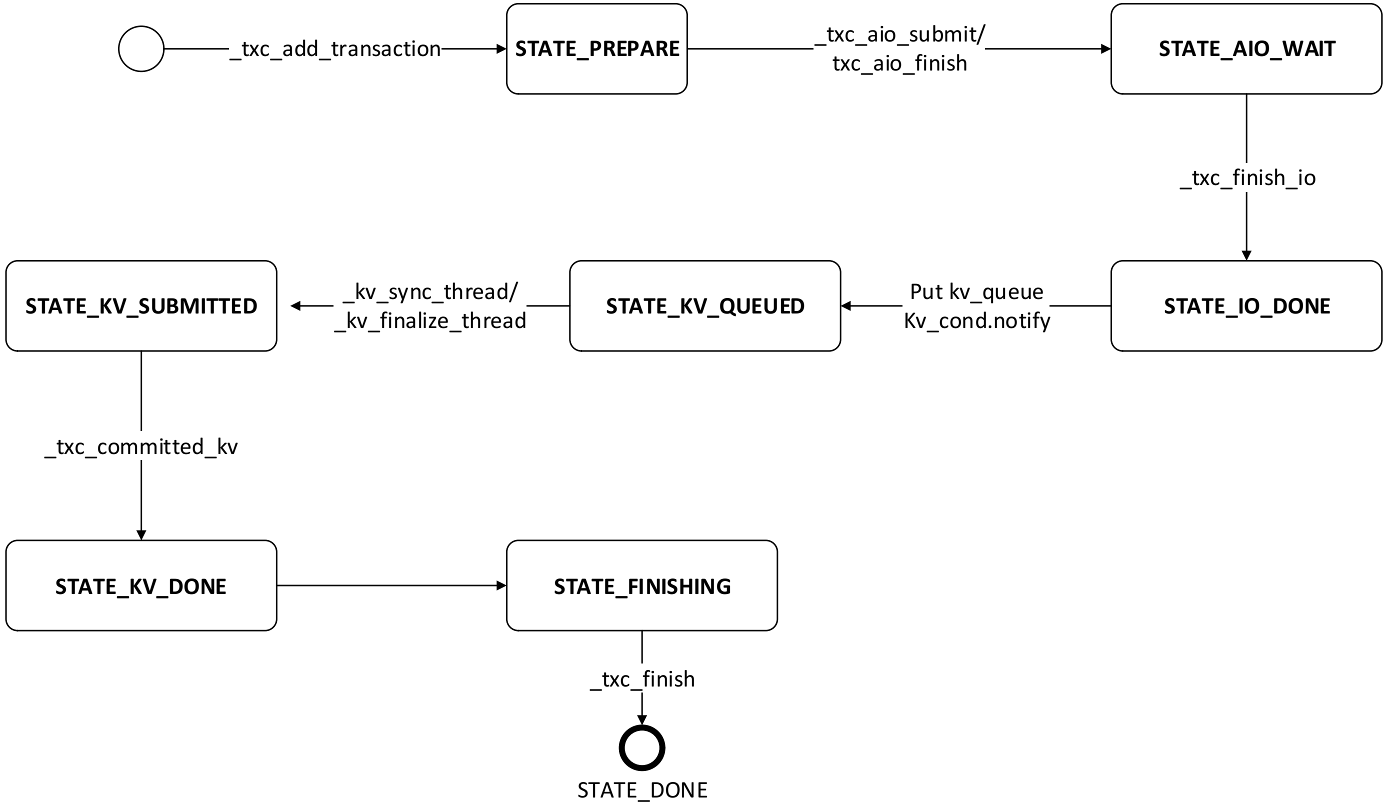
BlueStore – small IO (rewrite)

- Key/Value DB acts as WAL (deferred IO).
- Data is written to KV db, and return to upper layer.
- Later data is written into block device.
- Deferred IO entry is deleted from KV db.



BlueStore – big IO

- Key/Value DB acts as WAL (deferred IO).
- Data is written to KV db, and return to upper layer.
- Later data is written into block device.
- Deferred IO entry is deleted from KV db.



Agenda

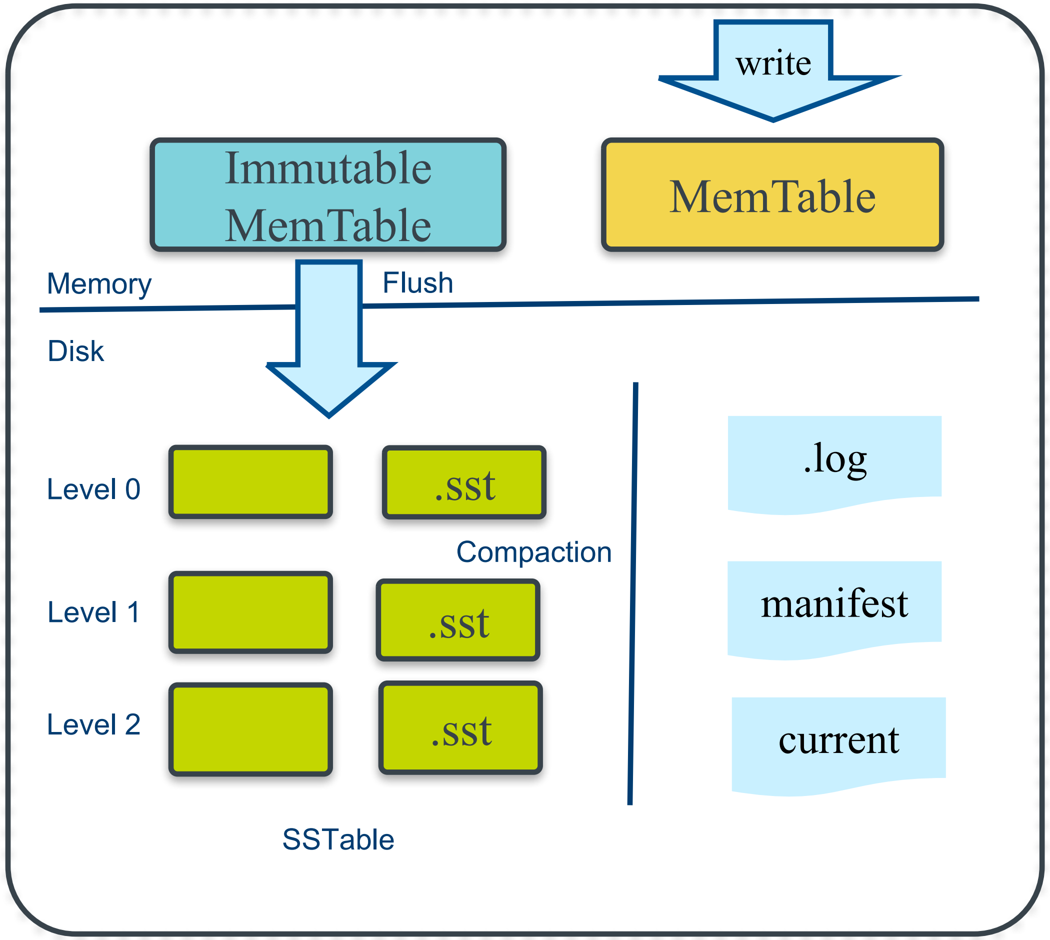
- BlueStore overview
- **Rocksdb overview**
- BlueStore latency over OSD
- Rocksdb optimization

Rocksdb

- A key-value database, originated by Google, improved by Facebook.
- Based on LSM (Log-Structure merge Tree).
- Flush
- Compaction

- Write: write into memTable
- Read: memTable fist, and then Level 0, Level 1 etc until finding the value.

- Write amplification
- Read amplification
- Space amplification

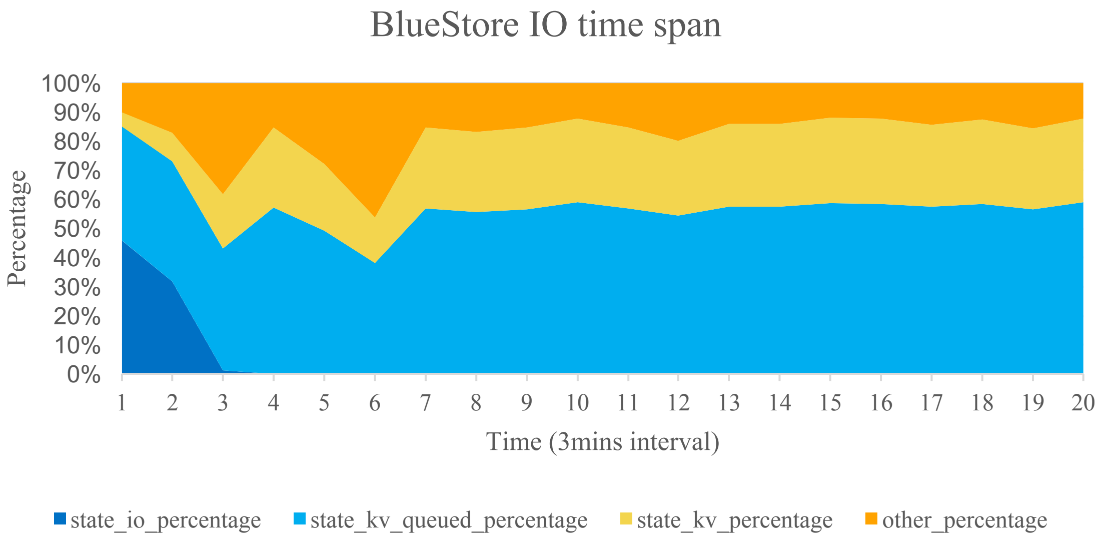


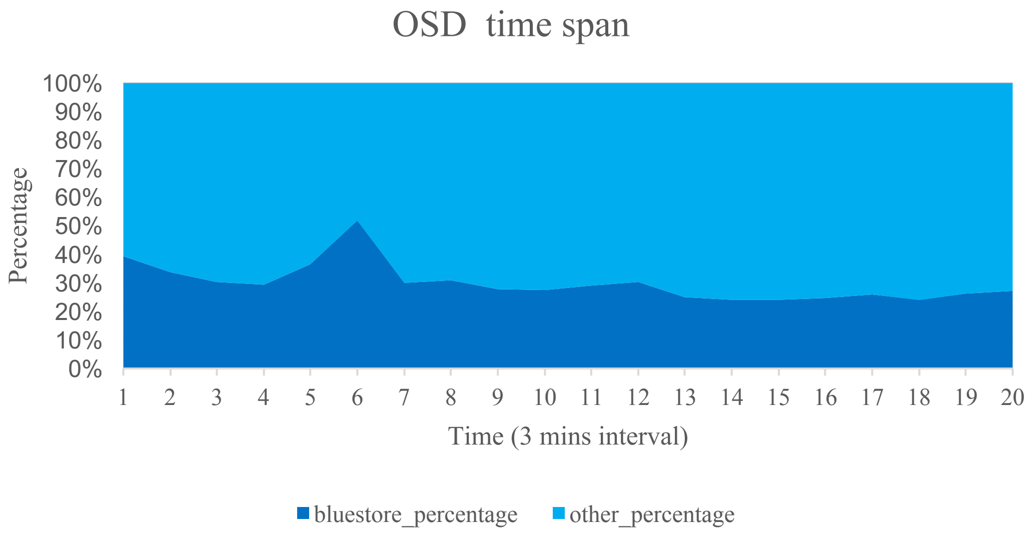
Agenda

- BlueStore overview
- Rocksdb overview
- **BlueStore latency over OSD**
- Rocksdb optimization

rbd10_sata_dev_nvme_db_4k

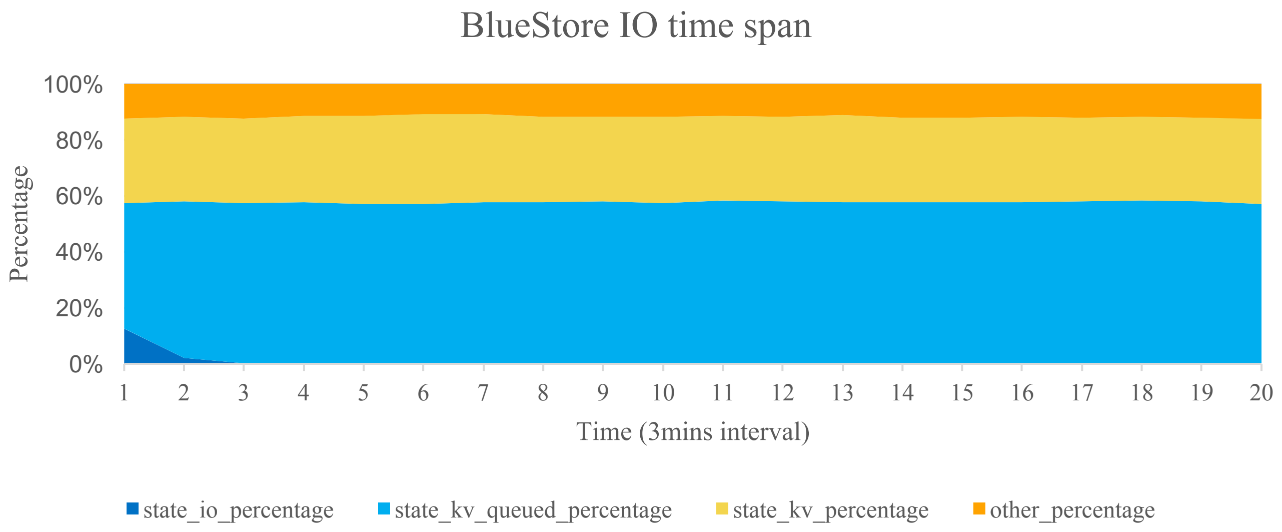
- Fio+librbd on 10 rbd images.
- 4k random write.
- Use Intel P3700 as db+wal, Intel S3520 as block device.

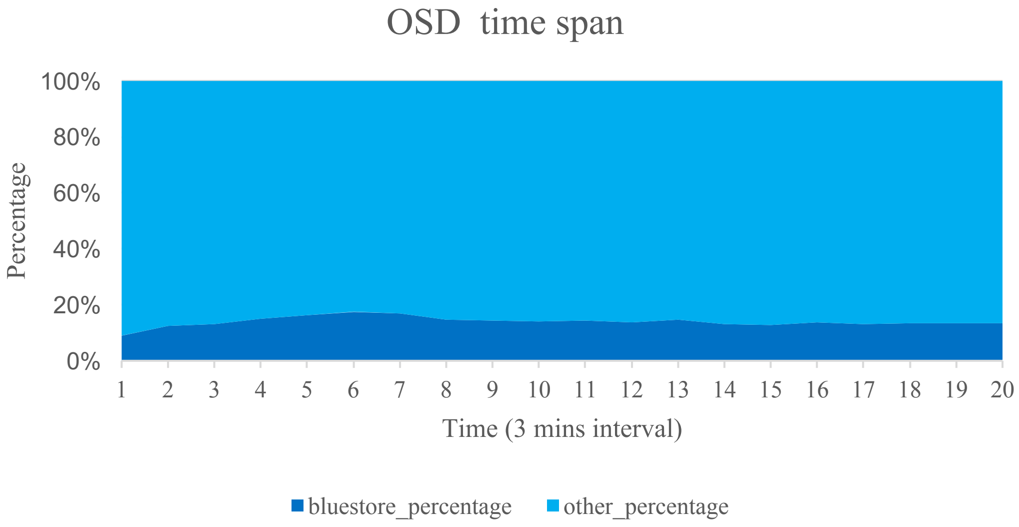




rbd10_nvme_all_4k

- Fio+librbd on 10 rbd images.
- 4k random write.
- Use Intel P3700 for all.



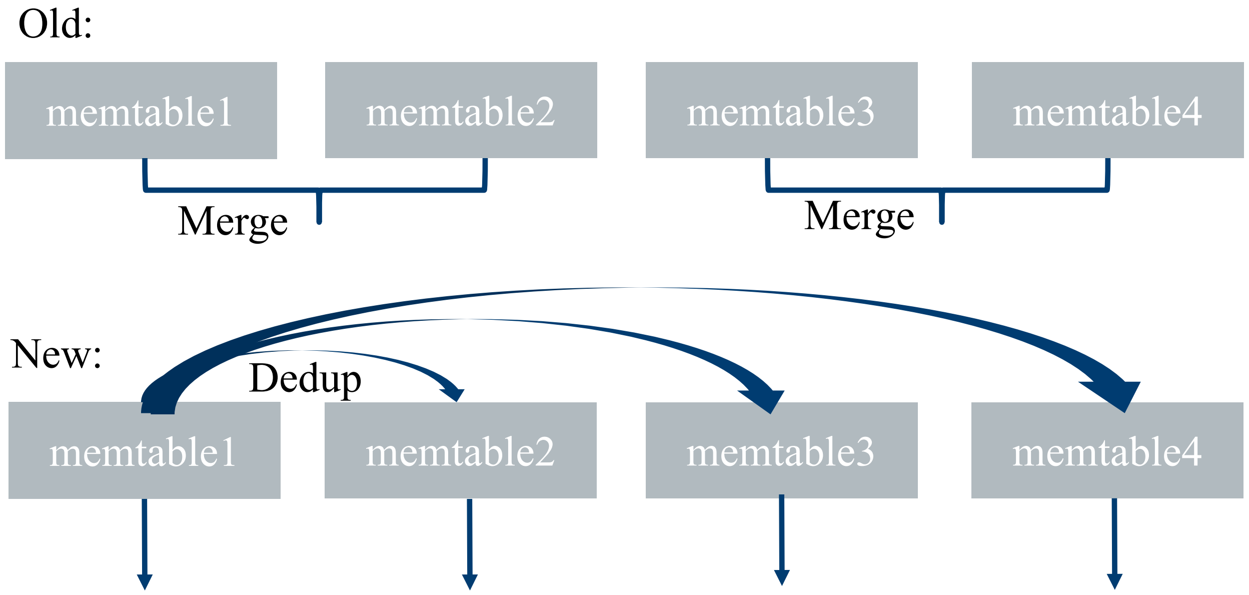


Agenda

- BlueStore overview
- Rocksdb overview
- BlueStore latency over OSD
- **Rocksdb optimization**

Rocksdb Optimization

- New merge style.
  - ➢ Merge key/value pairs recursively.
  - ➢ Decrease the data flushed into disks.



Old:

memtable1 memtable2 memtable3 memtable4

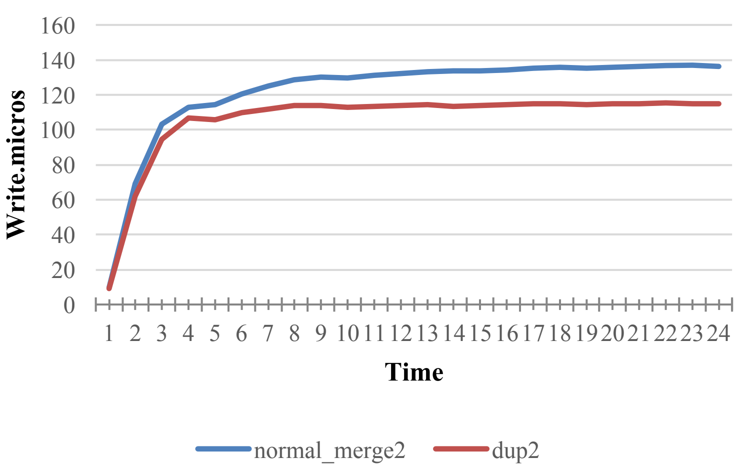
Merge Merge

New:

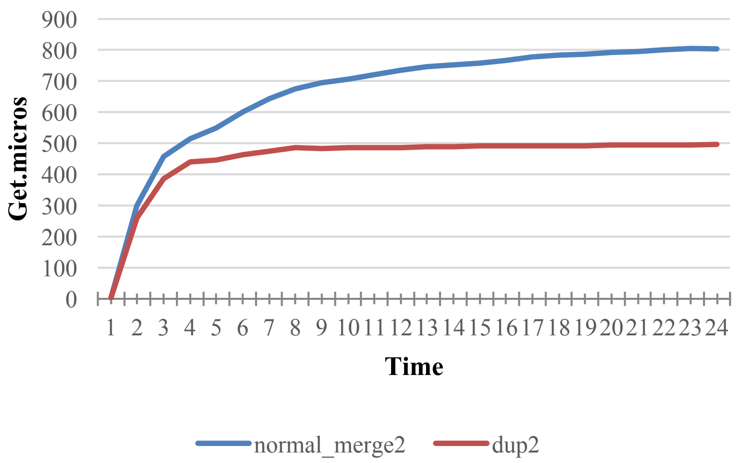
Dedup

memtable1 memtable2 memtable3 memtable4

db.write.micros

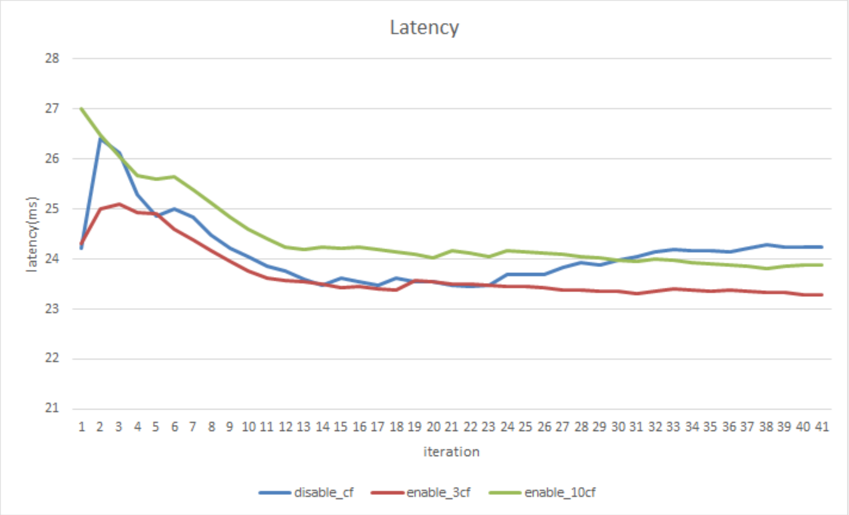


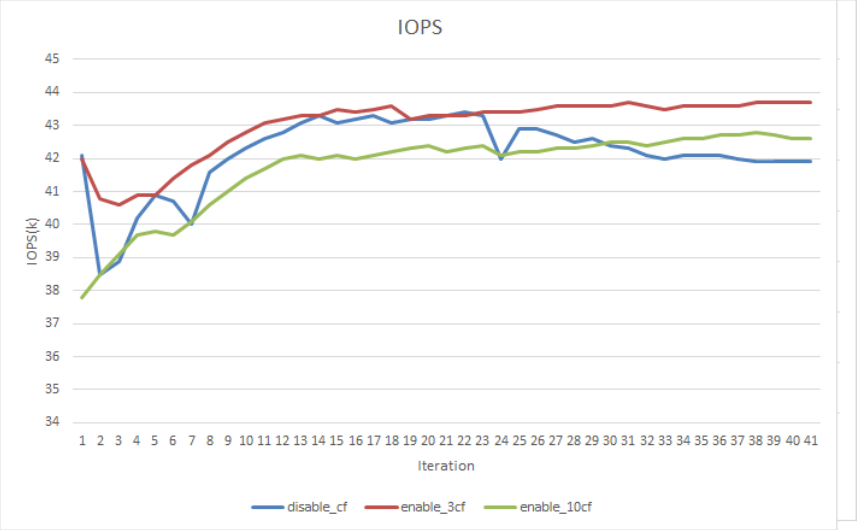
db.Get.micros



Rocksdb Optimization – cons.

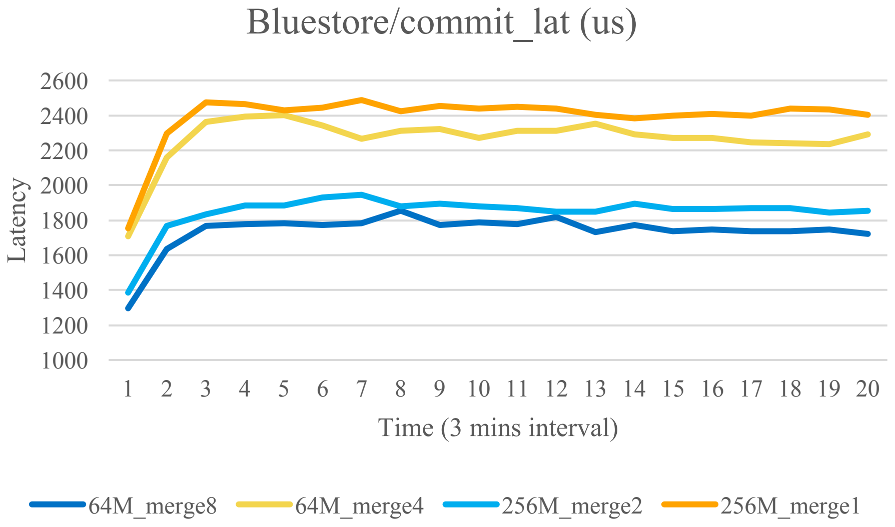
- Enable column families.
  - ➢ Create different column families for different kinds of metadata.
  - ➢ Set different options based on attributes of each type of metadata.
    - Omap
    - Deferred Ios
    - Other
  - ➢ This is first step. Further optimization based on cf is in progress.

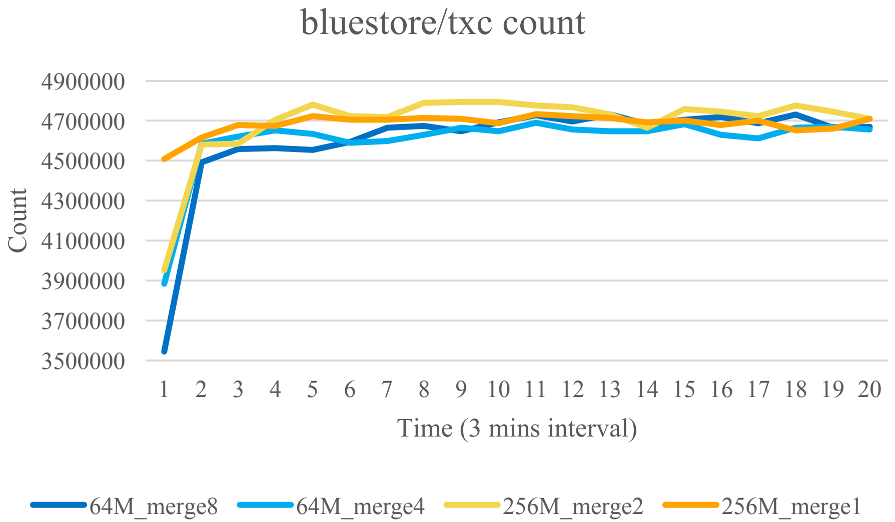




Impact of write buffer size

- Write_buffer_size
- Use Intel P3700
- 4k random Ios
- Different write_buffer_size and min_write_buffer_number_to_merge.





intel®