Understanding BlueStore

Ceph’s New Storage Backend

Tim Serong
Senior Clustering Engineer
SUSE
tserong@suse.com

Introduction to Ceph (the short, short version)

Ceph Provides

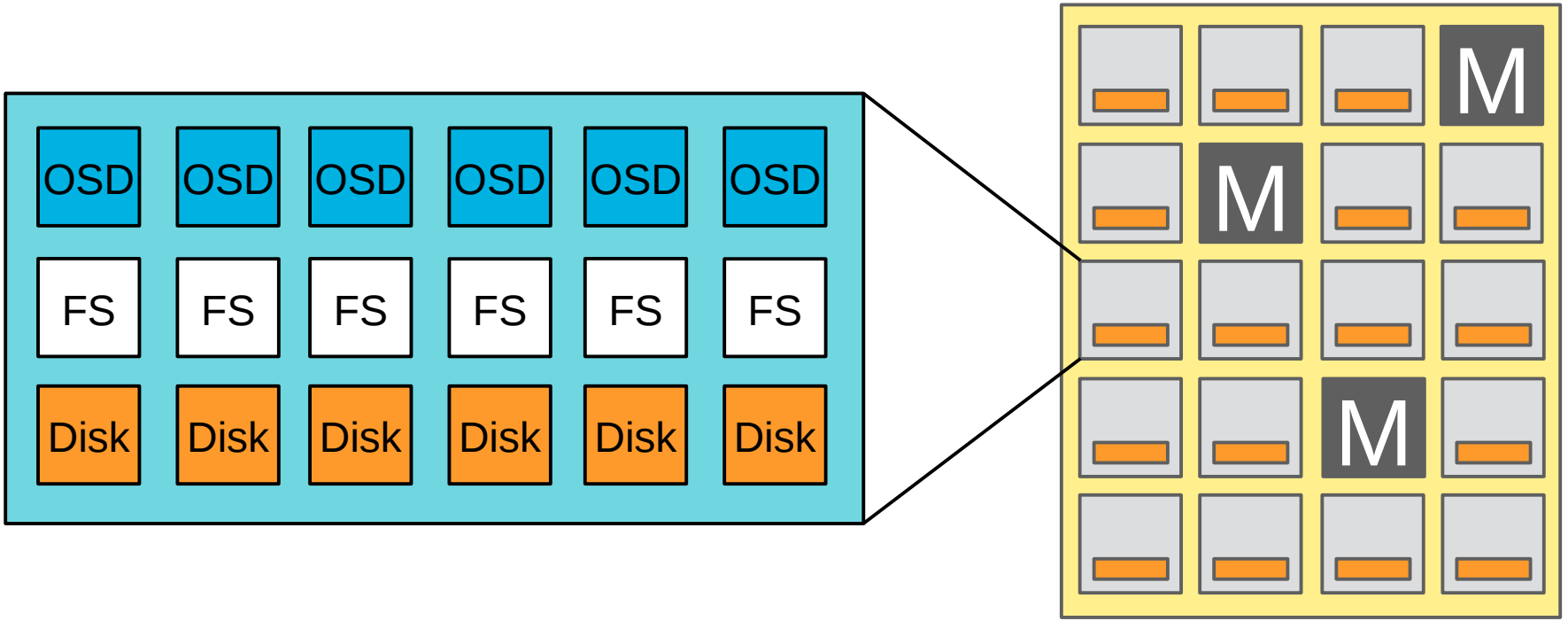
- **A resilient, scale-out storage cluster**
  - On commodity hardware
  - No bottlenecks
  - No single points of failure

- **Three interfaces**
  - Object (radosgw)
  - Block (rbd)
  - Distributed Filesystem (cephfs)

Internally, Regardless of Interface

- **All data stored as “objects”**
  - Aggregated into Placement Groups
- **Objects have:**
  - Data (byte stream)
  - Attributes (key/value pairs)
- **Objects live on Object Storage Devices**
  - Managed by Ceph Object Storage Daemons
- **An OSD is a disk, in a server**

More Internally



Even *More* Internally

- **FileStore**
  - XFS (previously ext4, btrfs)
  - Placement Group → directory
  - Object → file
  - LevelDB for key/value data

```
/var/lib/ceph/osd/ceph-0/
    current/
        0.1_head/
            object1
            object12
        0.7_head/
            object3
            object5
        0.a_head/
            object4
            object6
        meta/
            # OSD maps
        omap/
            # leveldb files
```

Problems with FileStore

Double Writes

- **No atomic write/update on POSIX filesystems**
- **Solution: Ceph Journal**
  - Write → OSD journal (O_DIRECT)
  - Write → OSD filesystem (buffered)
  - Sync filesystem, trim journal
- **Throughput halved**

Enumeration Sucks

- **Objects distributed by 32-bit hash**
- **Enumerated for scrub, rebalance etc.**
- **POSIX readdir() isn't ordered**
- **Directory tree with hash-value prefix**
  - Split when > ~100 files
  - Merged when < ~50 files
- **Complicated, odd performance issues**

```
...
/DIR_1
/DIR_1/DIR_0/obj-xxxxxx01
/DIR_1/DIR_4/obj-xxxxxx41
/DIR_1/DIR_8/obj-xxxxxx81
/DIR_1/DIR_C/obj-xxxxxxC1
...
/DIR_A/DIR_1/obj-xxxxx1A
/DIR_A/DIR_5/obj-xxxxx5A
/DIR_A/DIR_9/obj-xxxxx9A
/DIR_A/DIR_D/obj-xxxxxDA
...
```

Other Problems

- CPU utilization
- Unpredictable filesystem flushing
- Still finding bugs/issues with FileStore...

Solutions with BlueStore

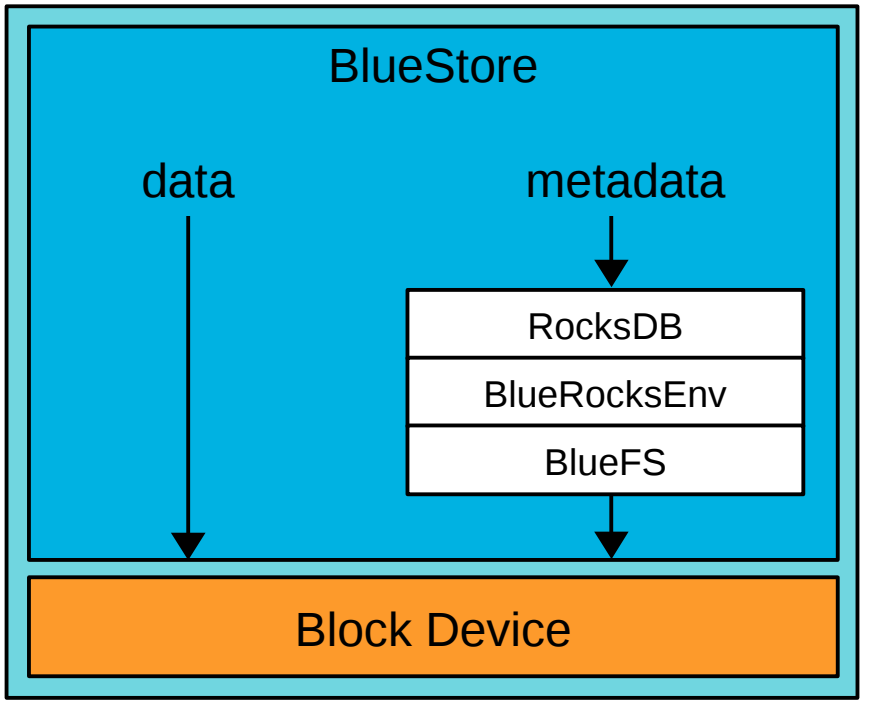
BlueStore = Block + NewStore

- Raw block devices
- RocksDB key/value store for metadata
- Data written direct to block device
- 2-3x performance boost over FileStore

Internally

- **All data and metadata on same block device by default**

- **Optionally can be split, e.g.:**
  - Data on HDD
  - DB on SSD
  - WAL on NVRAM



Bonus!

- **Pluggable block allocator**
- **Pluggable compression**
- **Checksums on reads**
- **RBD and CephFS usable with EC pools**

Availability

- Early prototype in Ceph Jewel (April 2016)
- Stable on-disk format in Ceph Kraken (January 2017)
- Recommended default in Ceph Luminous (RSN)
- Can co-exist with FileStore

Migration Strategies

Several Approaches

- **Fail in place**
  - Fail FileStore OSD
  - Create new BlueStore OSD on same device
- **Disk-wise replacement**
  - Create new BlueStore OSD on spare disk
  - Stop FileStore OSD on same host
- **Host-wise replacement**
  - Provision entire new host with BlueStore OSDs
  - Swap into old host’s CRUSH position
- **Evacuate and rebuild in place**
  - Evacuate FileStore OSD, fail when empty
  - Create new BlueStore OSD on same device

Several Approaches

- **Fail in place** → **period of reduced redundancy**
  - Fail FileStore OSD
  - Create new BlueStore OSD on same device
- **Disk-wise replacement** → **reduced *online* redundancy, requires extra drive slot**
  - Create new BlueStore OSD on spare disk
  - Stop FileStore OSD on same host
- **Host-wise replacement** → **no reduced redundancy, requires spare host**
  - Provision entire new host with BlueStore OSDs
  - Swap into old host's CRUSH position
- **Evacuate and rebuild in place** → **no reduced redundancy, requires free disk space**
  - Evacuate FileStore OSD, fail when empty
  - Create new BlueStore OSD on same device

Questions?


SUSE®
We adapt. You succeed.