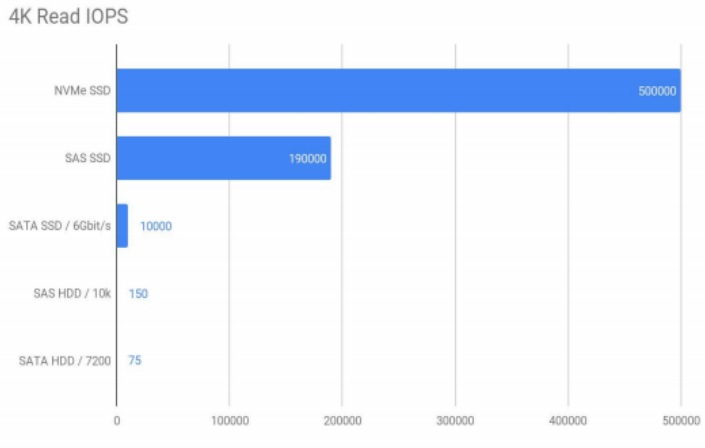
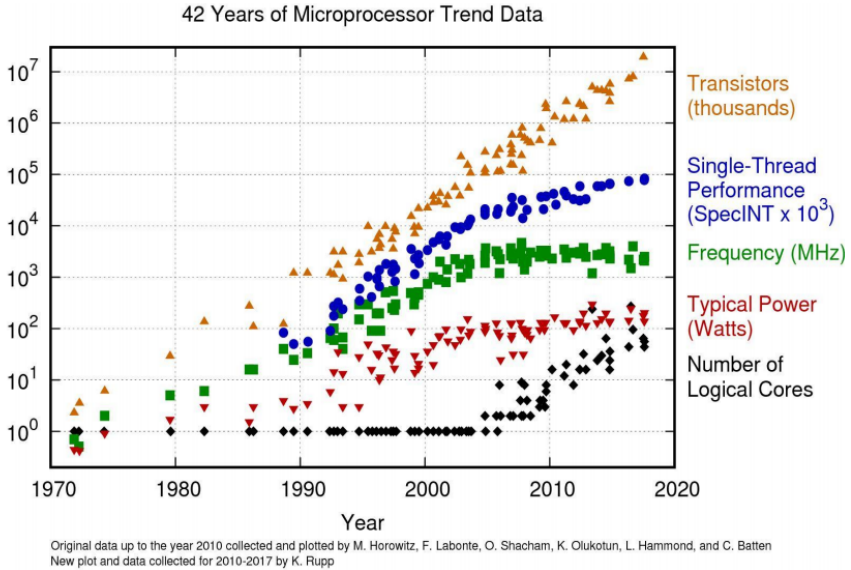
A New Replication Strategy in NVMe-oF Environment for Ceph

삼성전자, 손규호

CPU Not Getting Faster As Storage Does

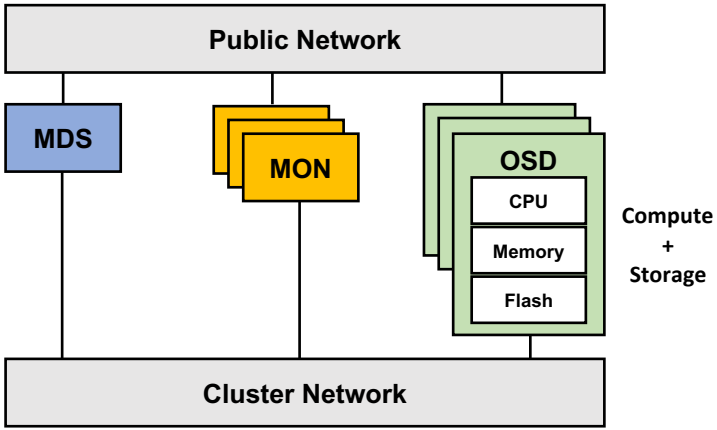
- But… Crimson targets fast networking & fast storage devices

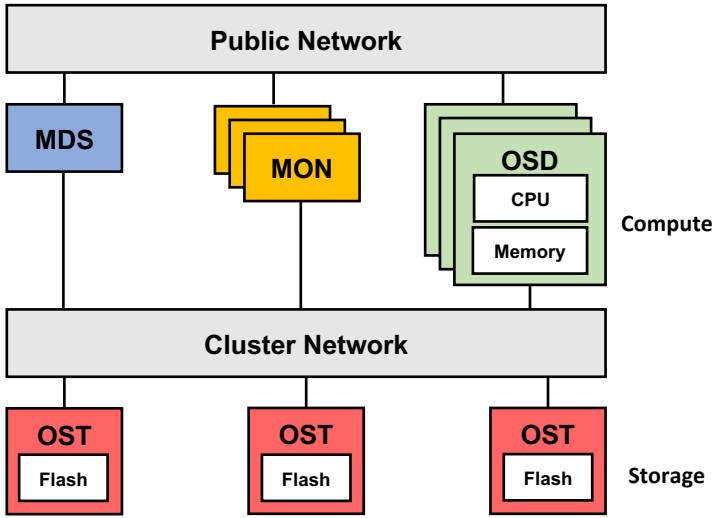




Trend in Industry, Storage Disaggregation

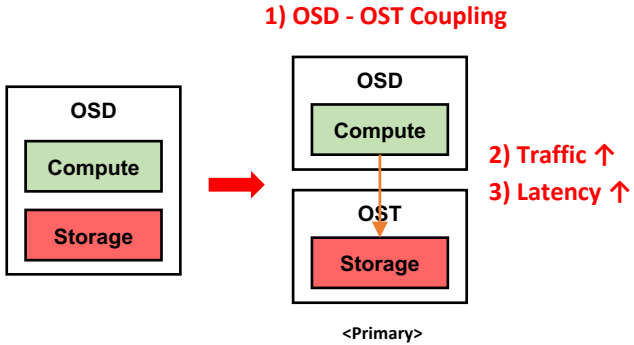
- Storage Disaggregation with NVMe-oF
  - Separates servers into compute and storage nodes
  - Any-to-any access among components
  - Independent resource scaling
  - NVMe-oF enables remote I/O operation with line speed





Storage Disaggregation in Ceph

- Ceph itself does not support storage disaggregation
  - Ceph does not aware of OST
  - OSD and OST is tightly coupled → Cannot share storage devices
  - Additional network traffic & latency
- Considerations to support storage disaggregation in Ceph
  - How to separate OSD roles into two node?
  - Which OSD manages object? Which OST stores data?
  - Any performance optimization?

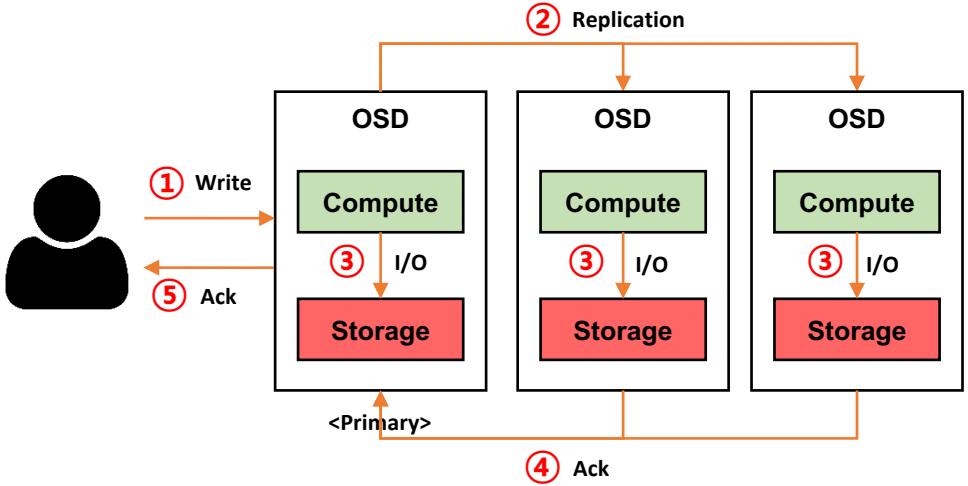


Ceph Write Operation

1. Client sends write request to the primary OSD
2. Primary OSD sends replication to secondary OSDs

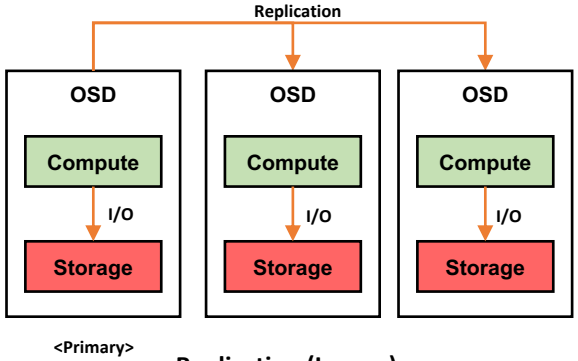
} Get OSD locations with CRUSH algorithm

3. Each OSD writes object data to its storage
4. When I/O completes, secondary OSDs send Ack to the primary OSD
5. Primary OSD sends Ack to the client

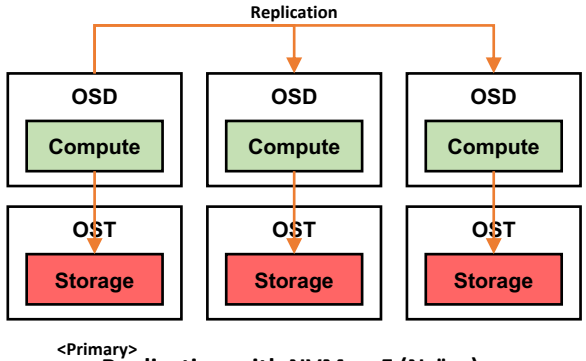


Approaches for Storage Disaggregation

*** We target systems where the CPU is bottleneck point (100Gbps↑, NVMe SSDs)**



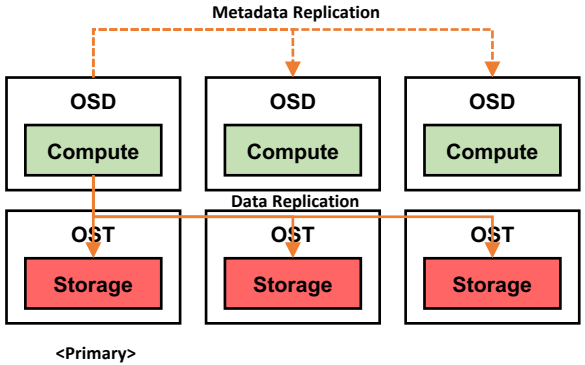
Replication (Legacy)



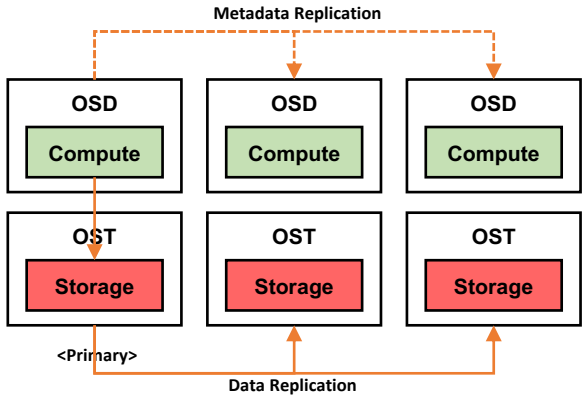
Replication with NVMe-oF (Naïve)

Network Overhead ↑

Network Overhead ↓



Replication with NVMe-oF (Optimized 1)

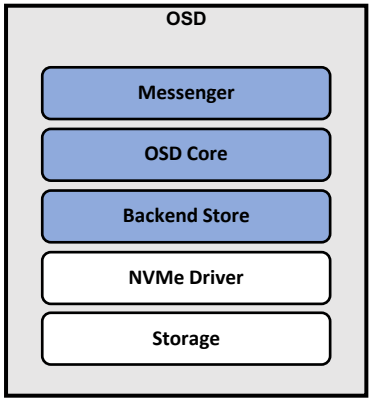


Replication with NVMe-oF (Optimized 2)

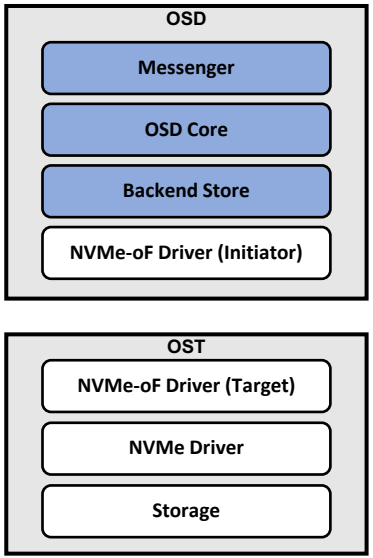
Network Overhead ↓
OSD CPU Consumption ↓
Implementation ↑

Consideration 1 – Separation of OSD

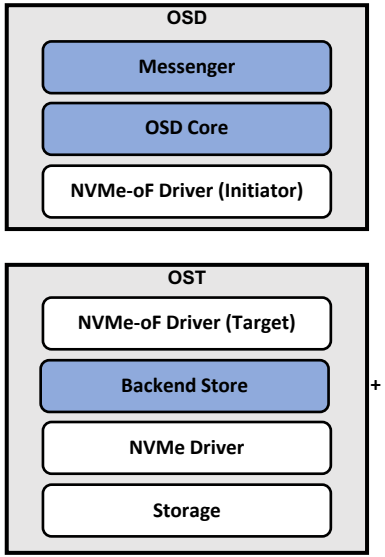
- OSD consists of OSD Core (control plane) and Backend Store (data plane)
  - OSD Core: object mapping, replication, recovery
  - Backend Store: data block management
- Backend Store should be located in OST to reduce data traffic



OSD Stack without NVMe-oF



OSD Stack with NVMe-oF (Naïve)



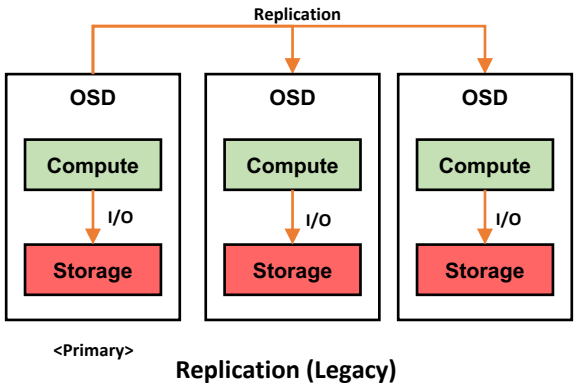
OSD Stack with NVMe-oF (Optimized)

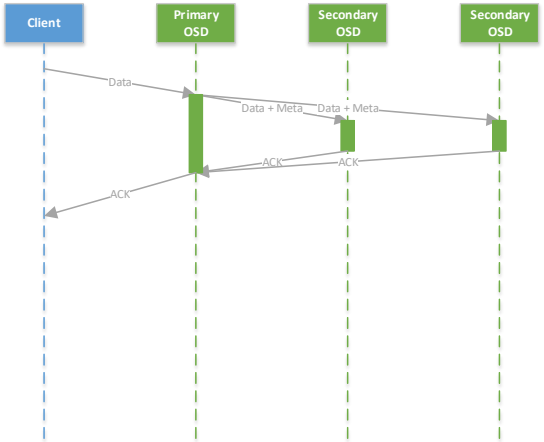
Consideration 2 – Data Traffic / Latency at Replication

Traffic: 3 Data Copy

Latency: 2 Data Copy + 2 ACK

Not Support Storage Disaggregation



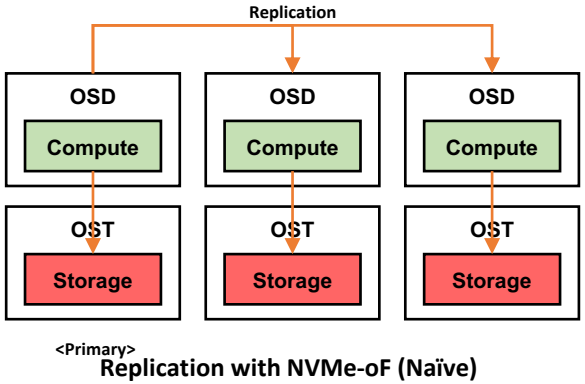


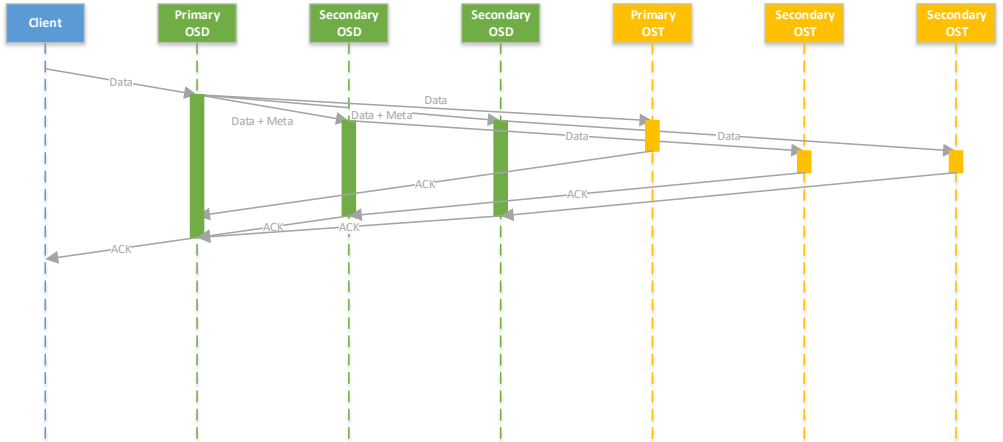
Consideration 2 – Data Traffic / Latency at Replication

Traffic: 6 Data Copy

Latency: 3 Data Copy + 3 ACK

No Ceph Modification, Double Network Traffic, Additional Latency



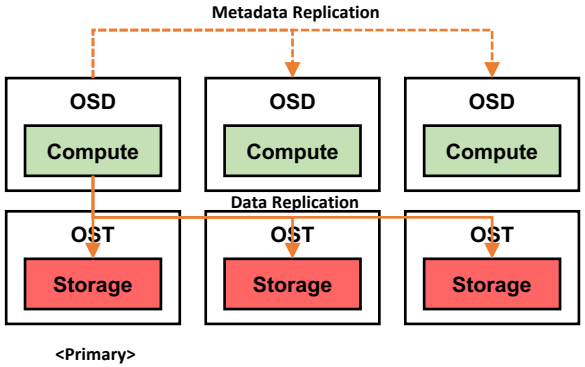


Consideration 2 – Data Traffic / Latency at Replication

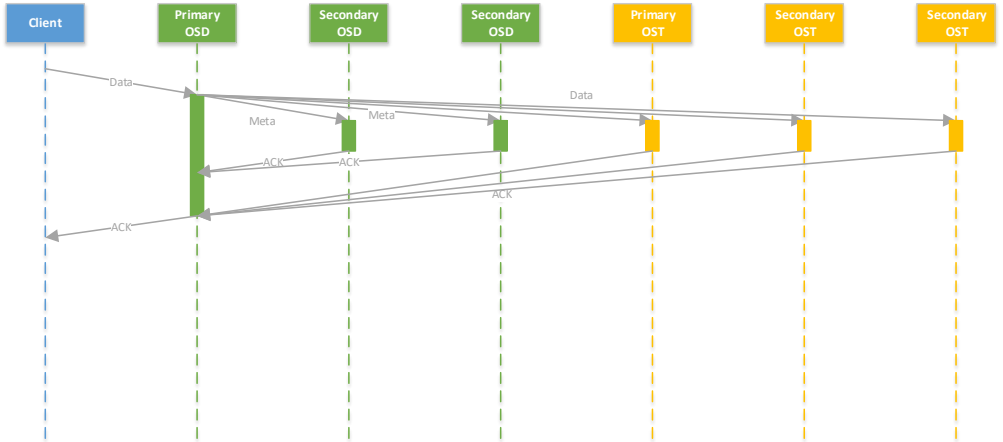
Traffic: 4 Data Copy

Latency: 2 Data Copy + 2 ACK

No Additional Latency, Primary OSD Works Hard



Replication with NVMe-oF (Optimized 1)

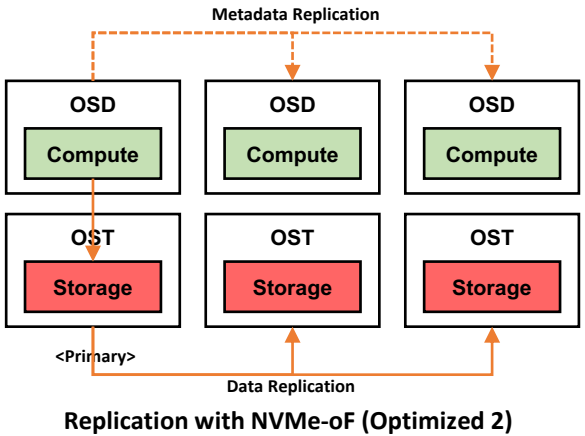


Consideration 2 – Data Traffic / Latency at Replication

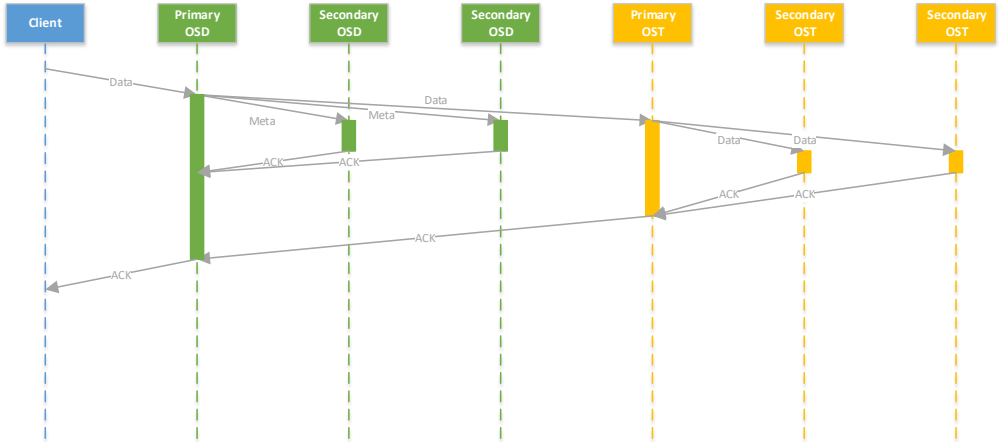
Traffic: 4 Data Copy

Latency: 3 Data Copy + 3 ACK

Loadbalance, Implementation of Replication Feature in OST, Additional Latency



Replication with NVMe-oF (Optimized 2)

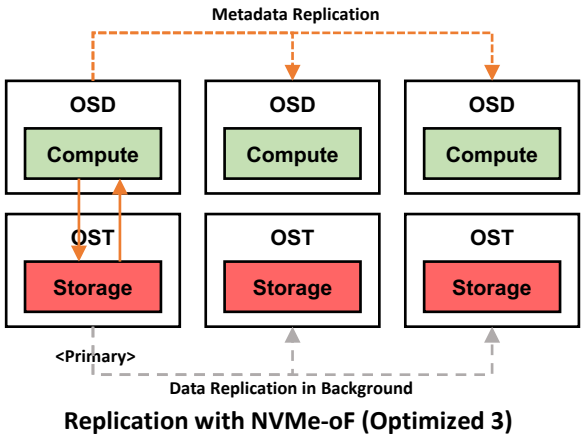


Consideration 2 – Data Traffic / Latency at Replication

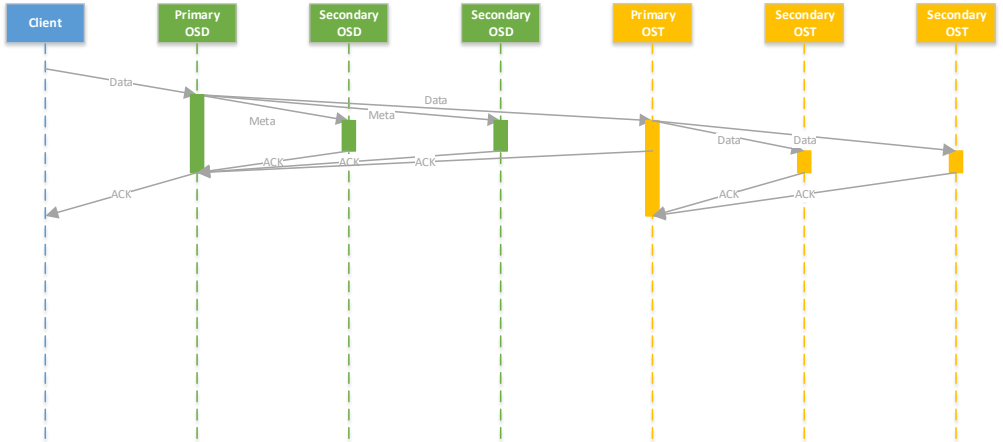
Traffic: 4 Data Copy

Latency: 2 Data Copy + 2 ACK

Replication in Background, Need Use of Persistent Memory

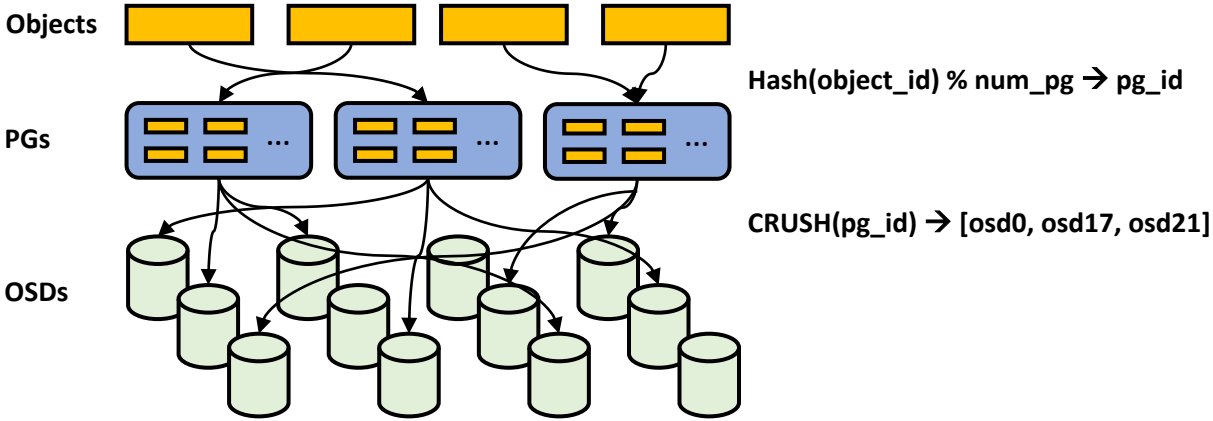


Replication with NVMe-oF (Optimized 3)

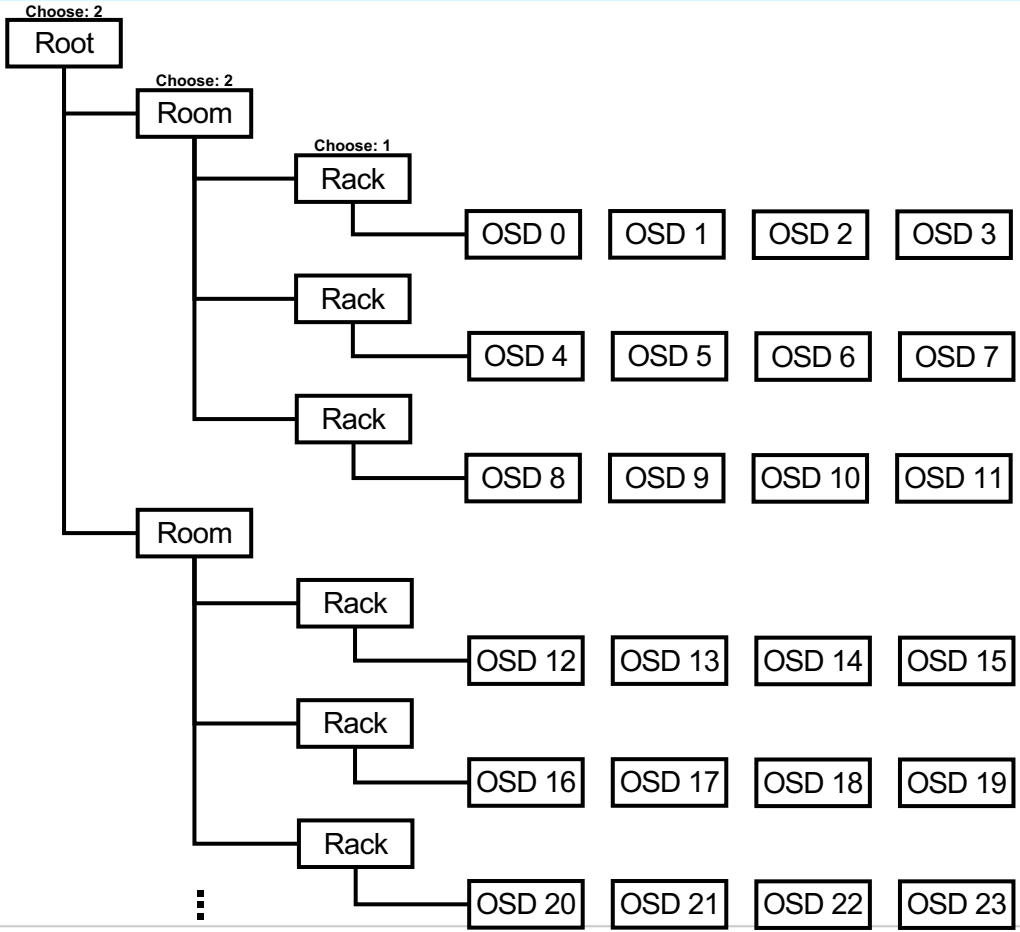


CRUSH – Distribute Data Considering Fault Domain

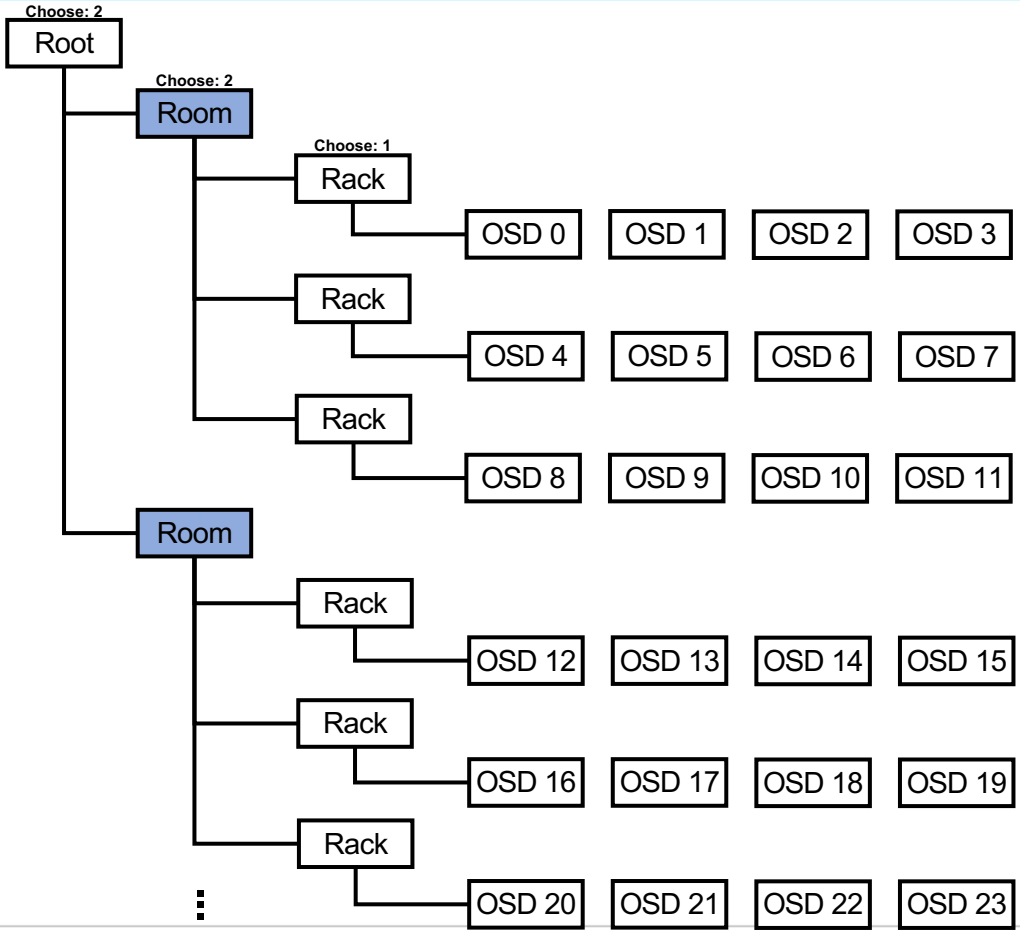
- CRUSH (Controlled, Scalable, Decentralized Placement of Replicated Data)
  - Data distribution algorithm designed for “dynamic” distributed storage system
  - Any party in a system can independently calculate the location of any object
  - Facilitate the addition and removal of storage while minimizing unnecessary data movement



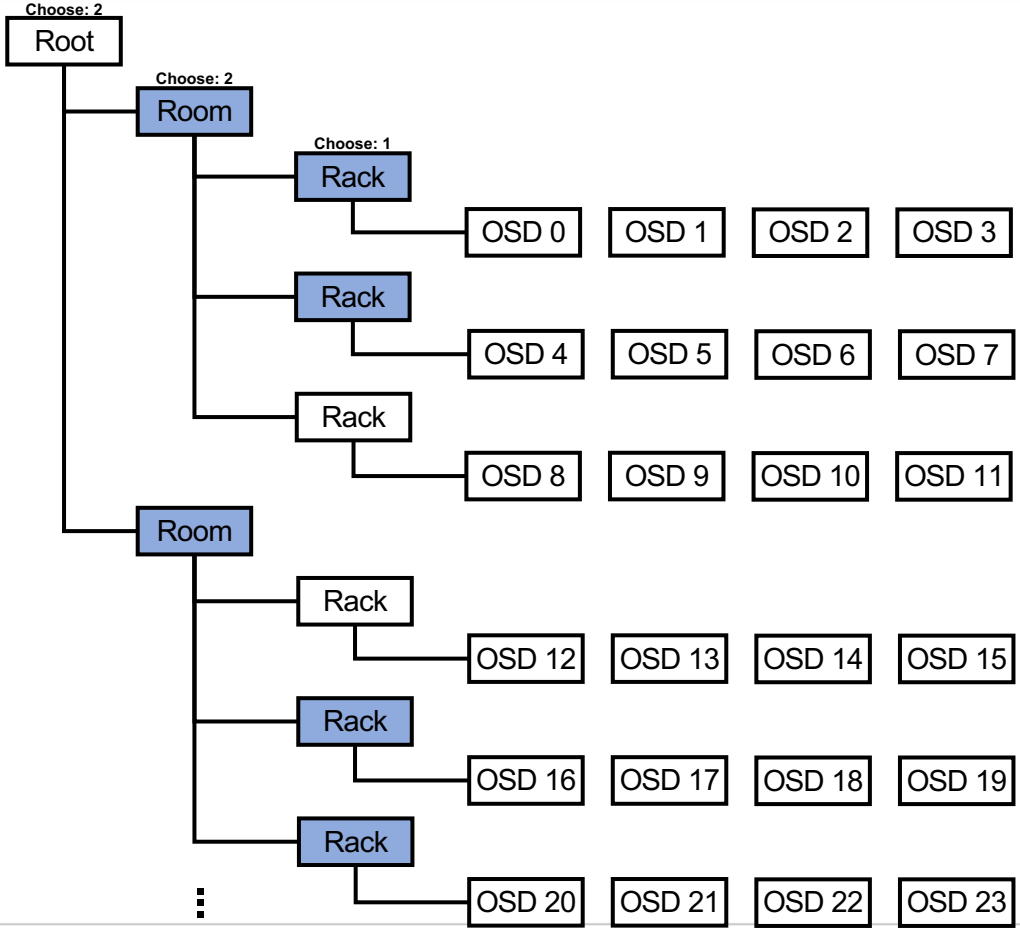
CRUSH – Distribute Data Considering Fault Domain



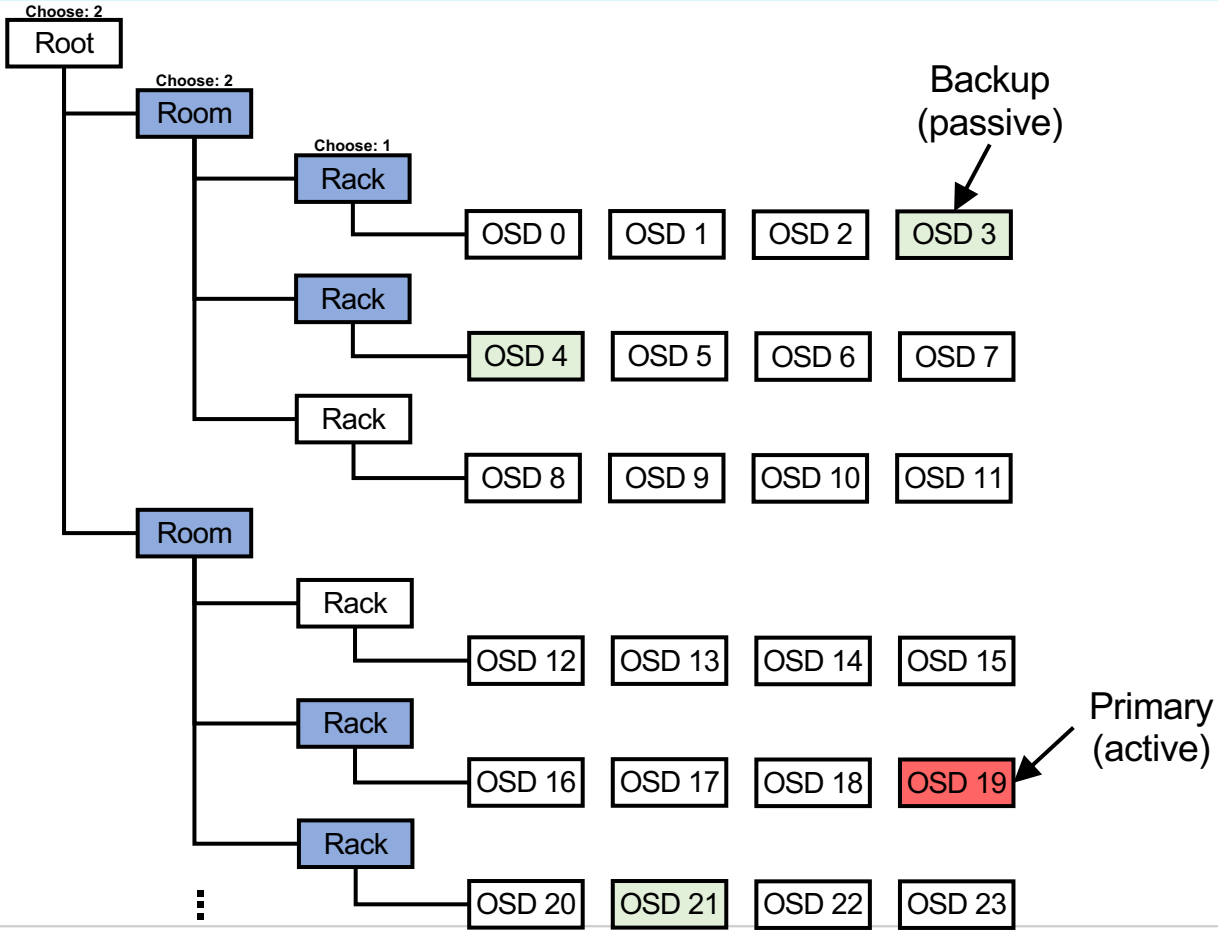
CRUSH – Distribute Data Considering Fault Domain



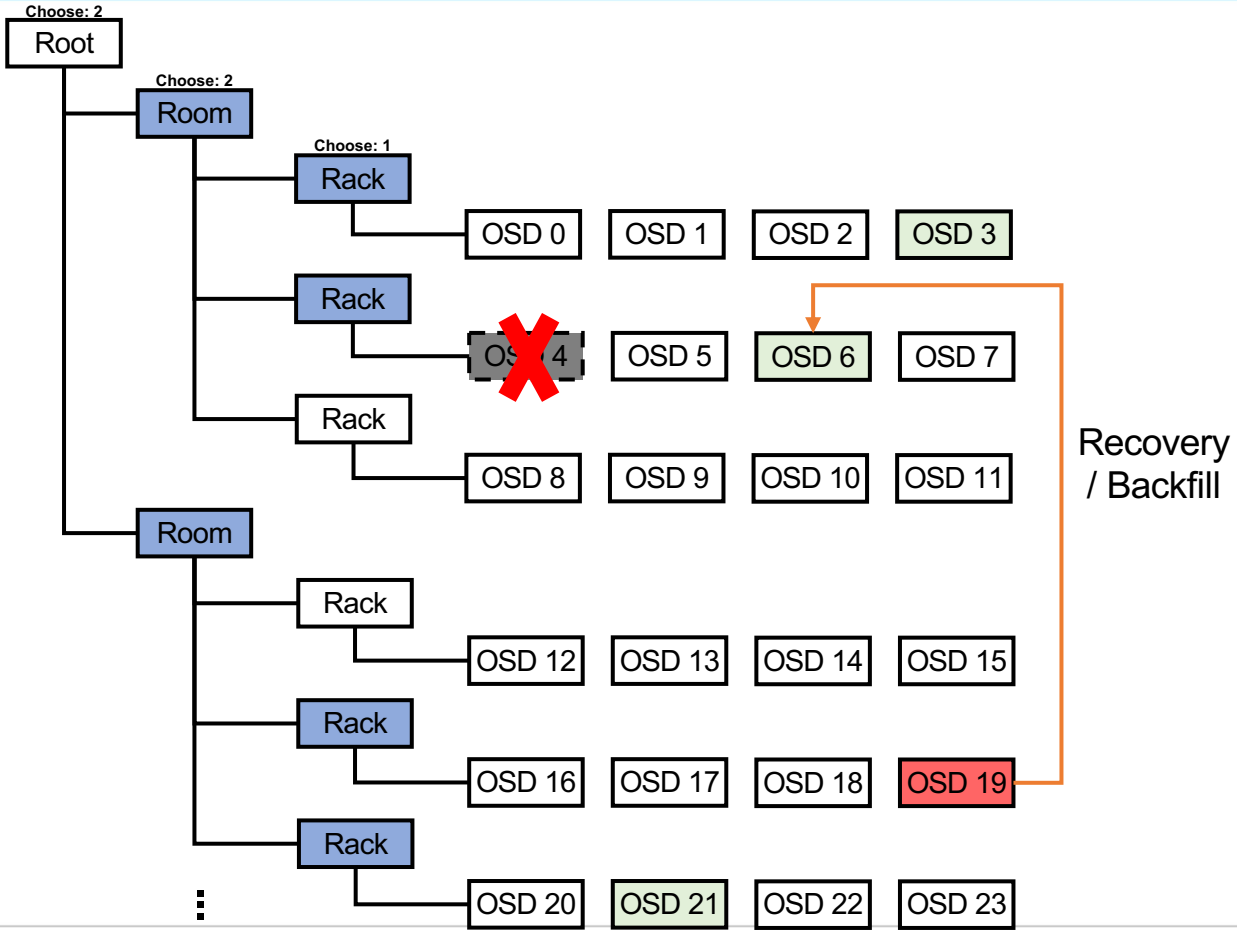
CRUSH – Distribute Data Considering Fault Domain



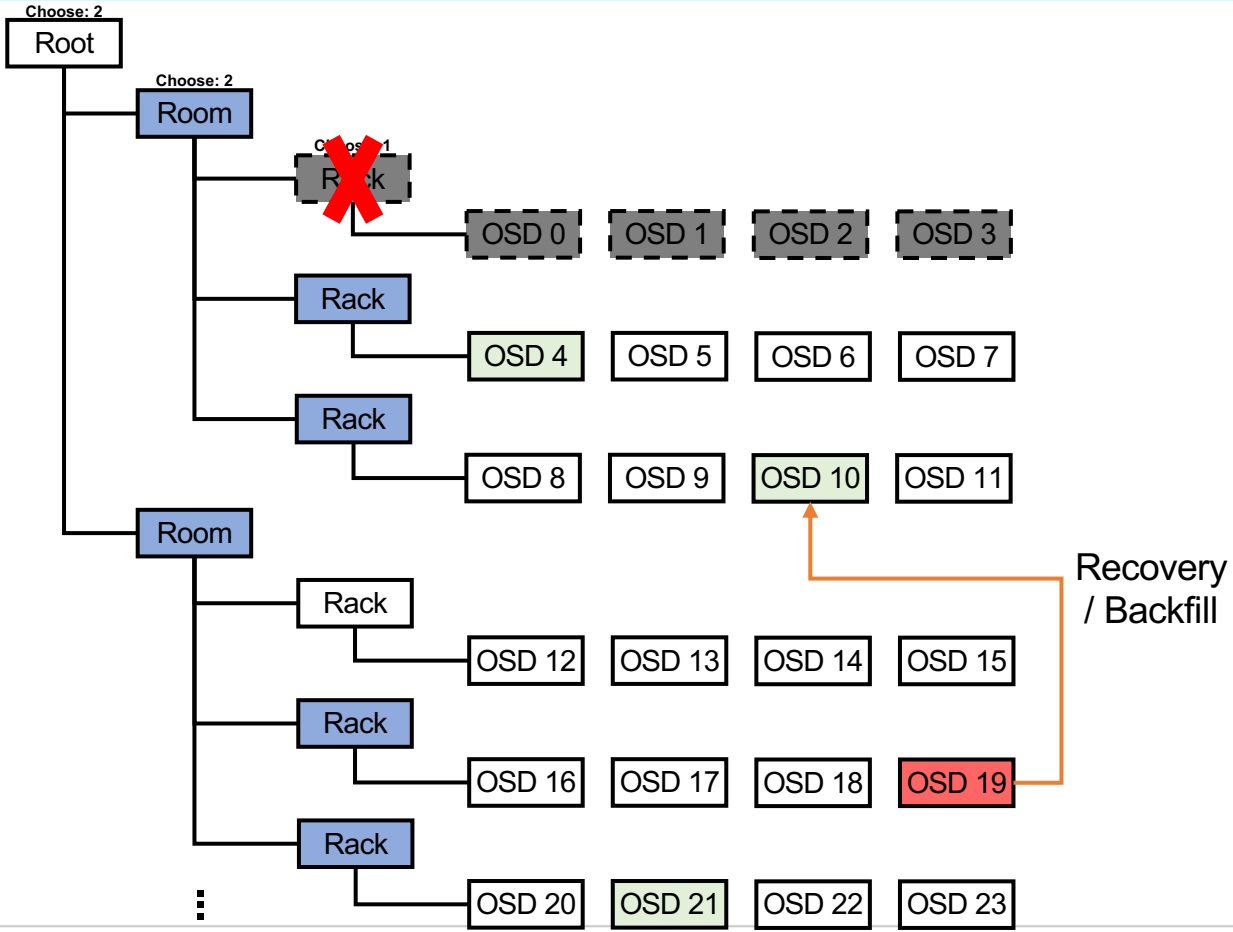
CRUSH – Distribute Data Considering Fault Domain



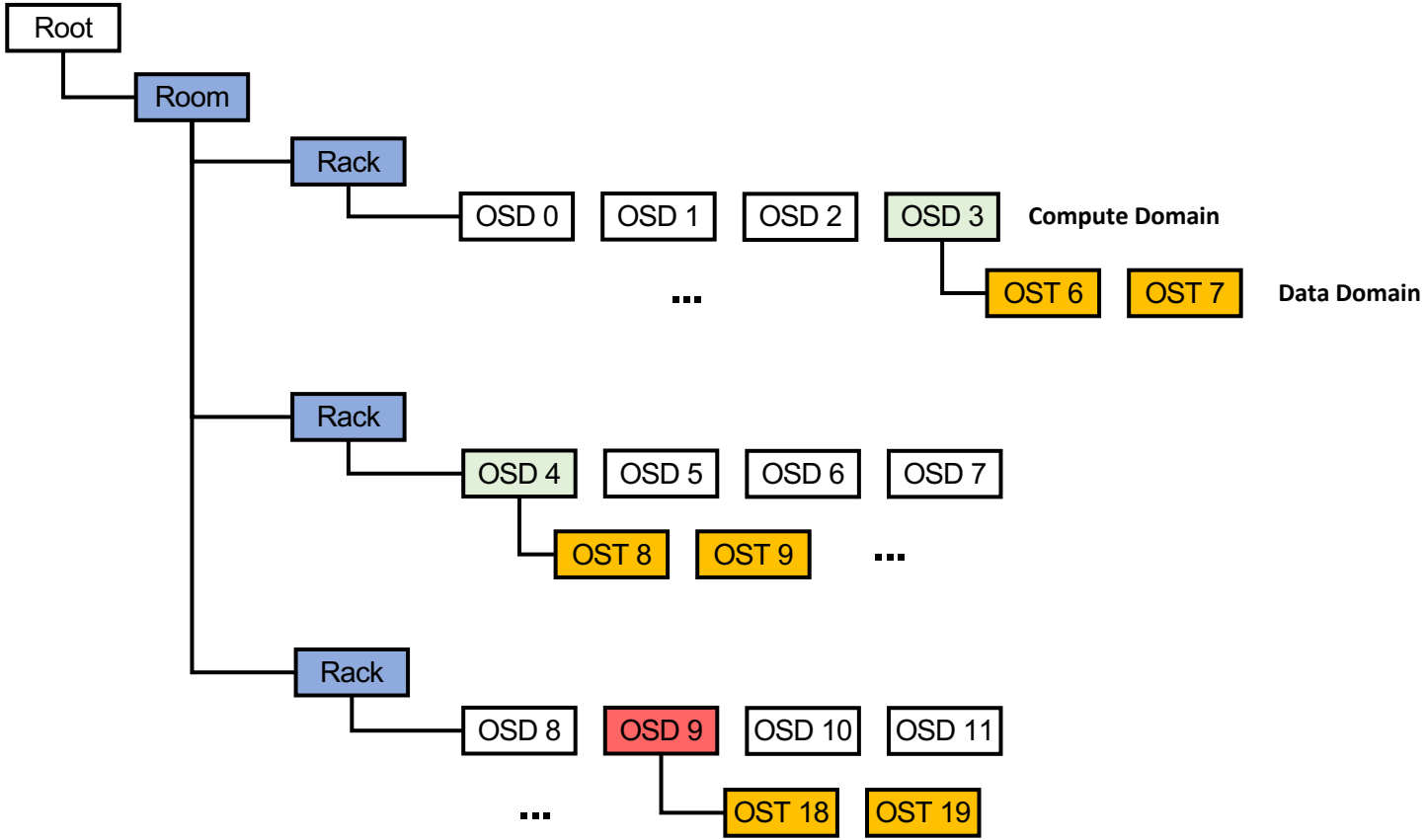
CRUSH – Distribute Data Considering Fault Domain



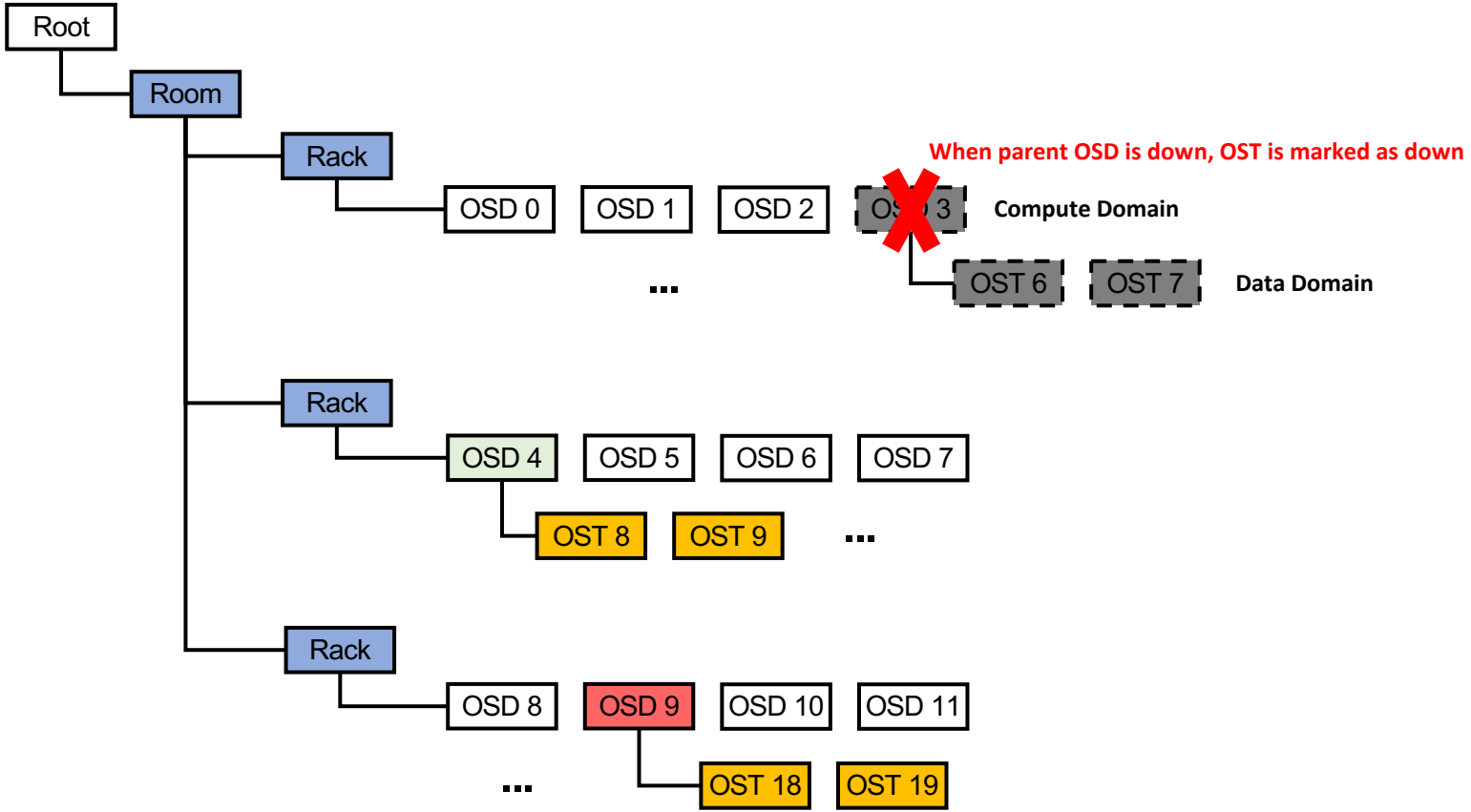
CRUSH – Distribute Data Considering Fault Domain



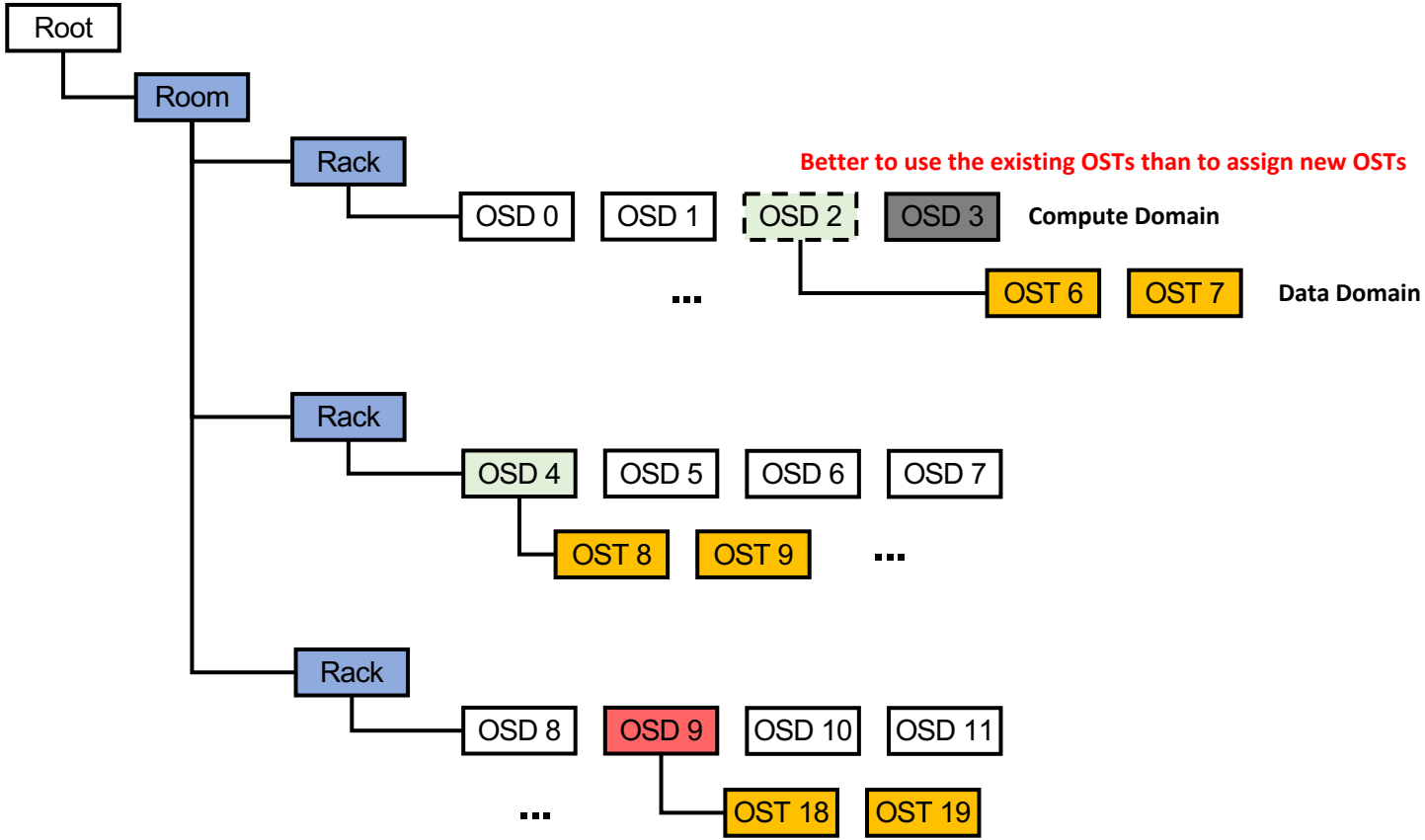
CRUSH – Distribute Data Considering Fault Domain



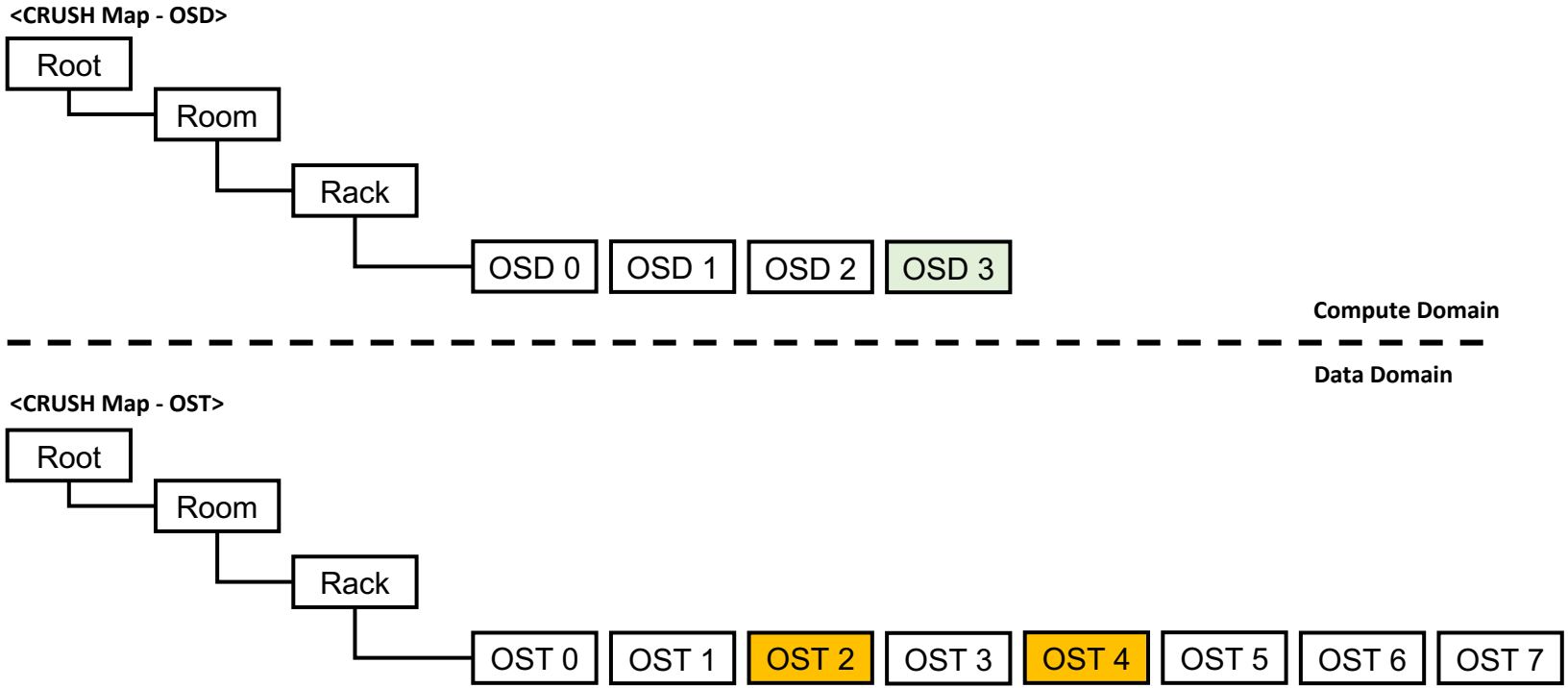
CRUSH – Distribute Data Considering Fault Domain



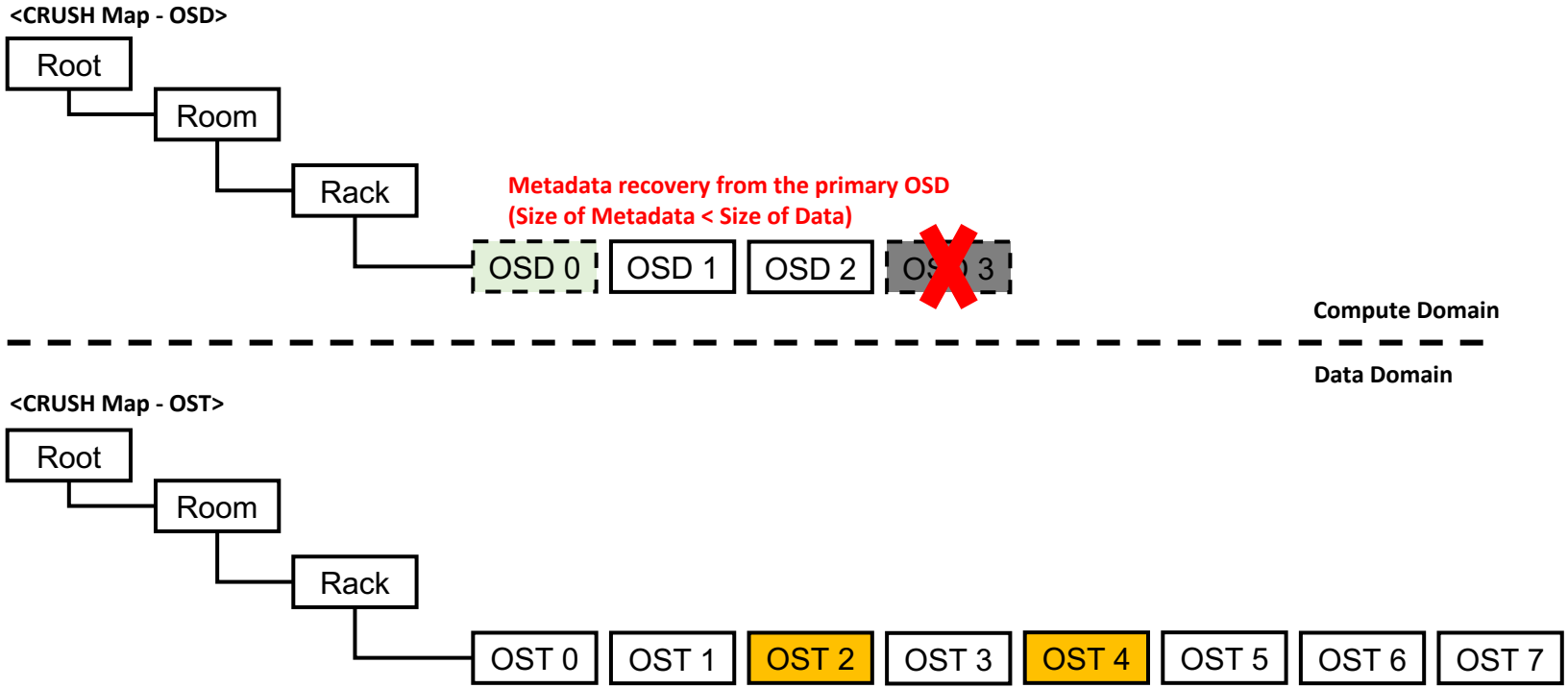
CRUSH – Distribute Data Considering Fault Domain



Consideration3 – Separating Compute & Data Domain



Consideration3 – Separating Compute & Data Domain



Summary

- Storage Disaggregation can bring new optimization opportunity to Ceph
- Modules related to block operation should be moved from OSD to OST
- Replication model influences a lot on storage performance
- Compute and data domain should be decoupled in CRUSH

SOLUTION

CORE VALUES

Speciality

Ownership

Leadership

Upgrowth

Together

Integrity

Openness

Now

Thank You