†

Understanding Write Behaviors of Storage Backends in Ceph Object Store

Dong-Yun Lee, Kisik Jeong, Sang-Hoon Han, Jin-Soo Kim,

Joo-Young Hwang† and Sangyeun Cho†

How Ceph Amplifies Writes

How Ceph Amplifies Writes

client

Data

Ceph metadata

Ceph journal

Describe data

Make updates atomic and consistent

How Ceph Amplifies Writes

client

Data

Ceph metadata

Ceph journal

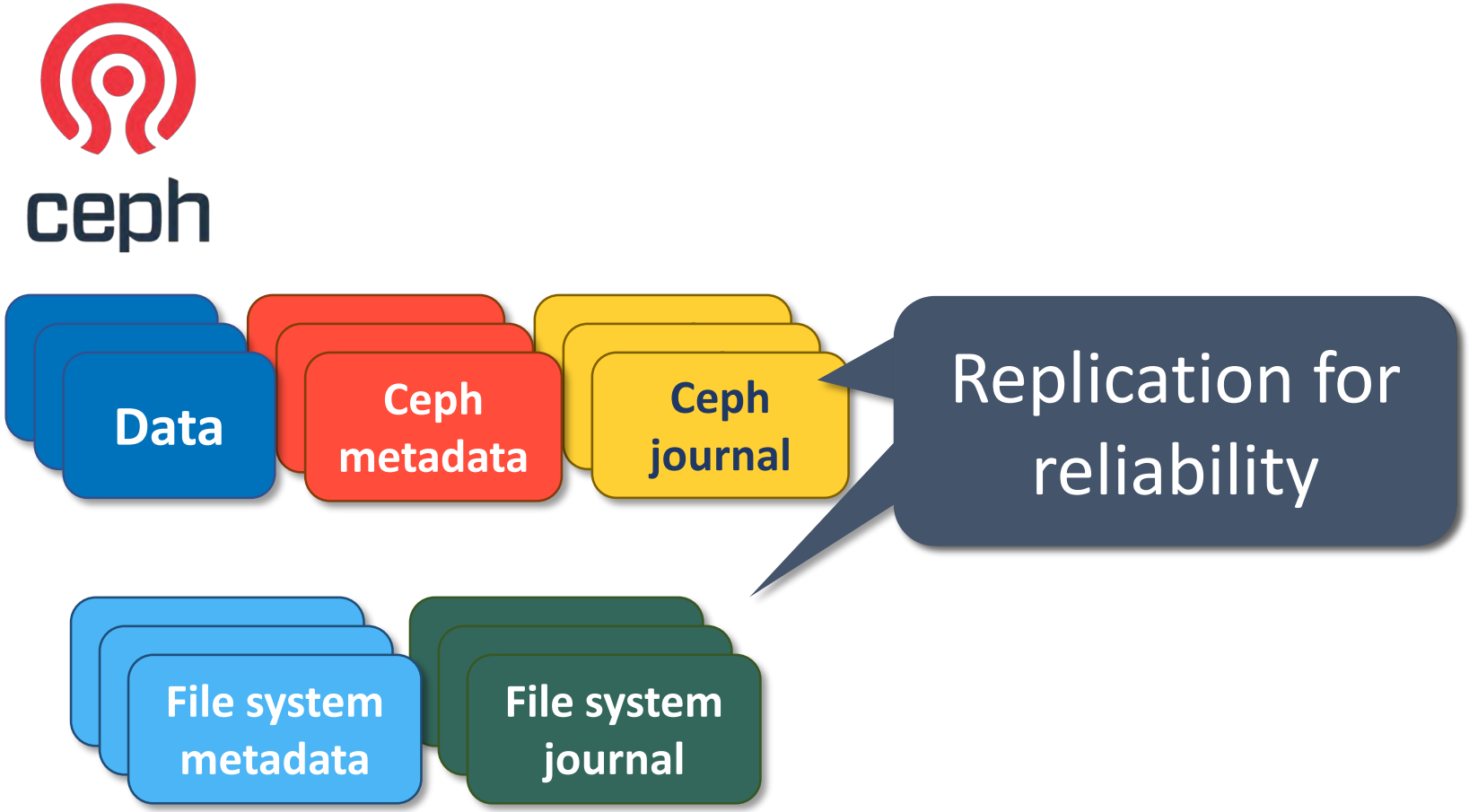
File system metadata

File system journal

By local file system (XFS)

How Ceph Amplifies Writes

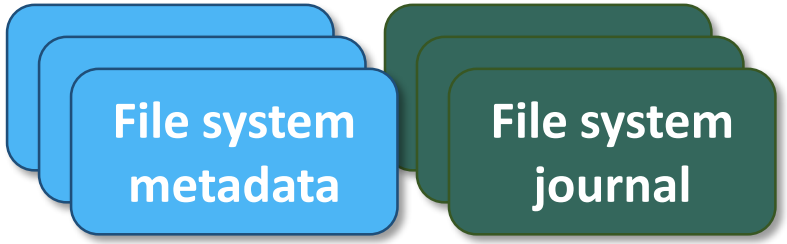
client



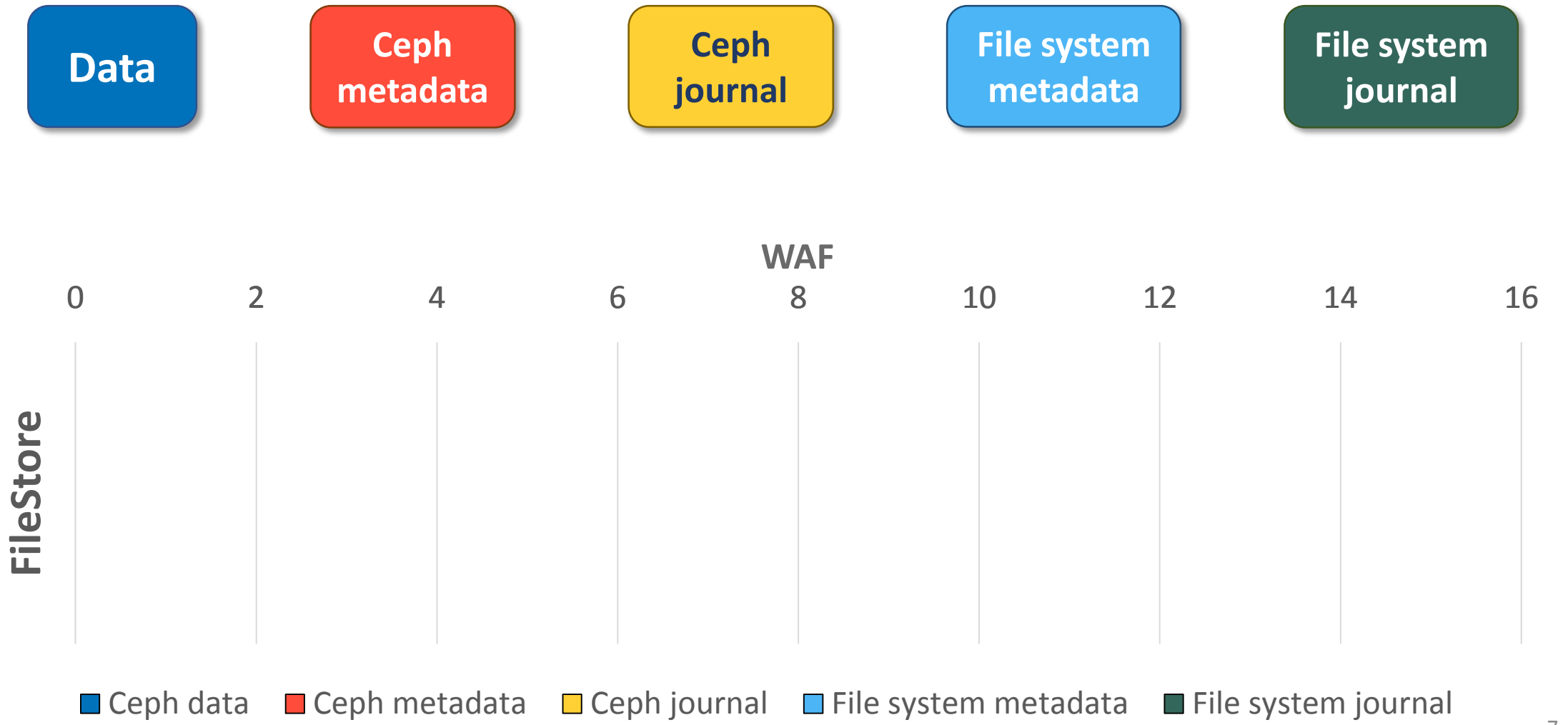
How Ceph Amplifies Writes

client

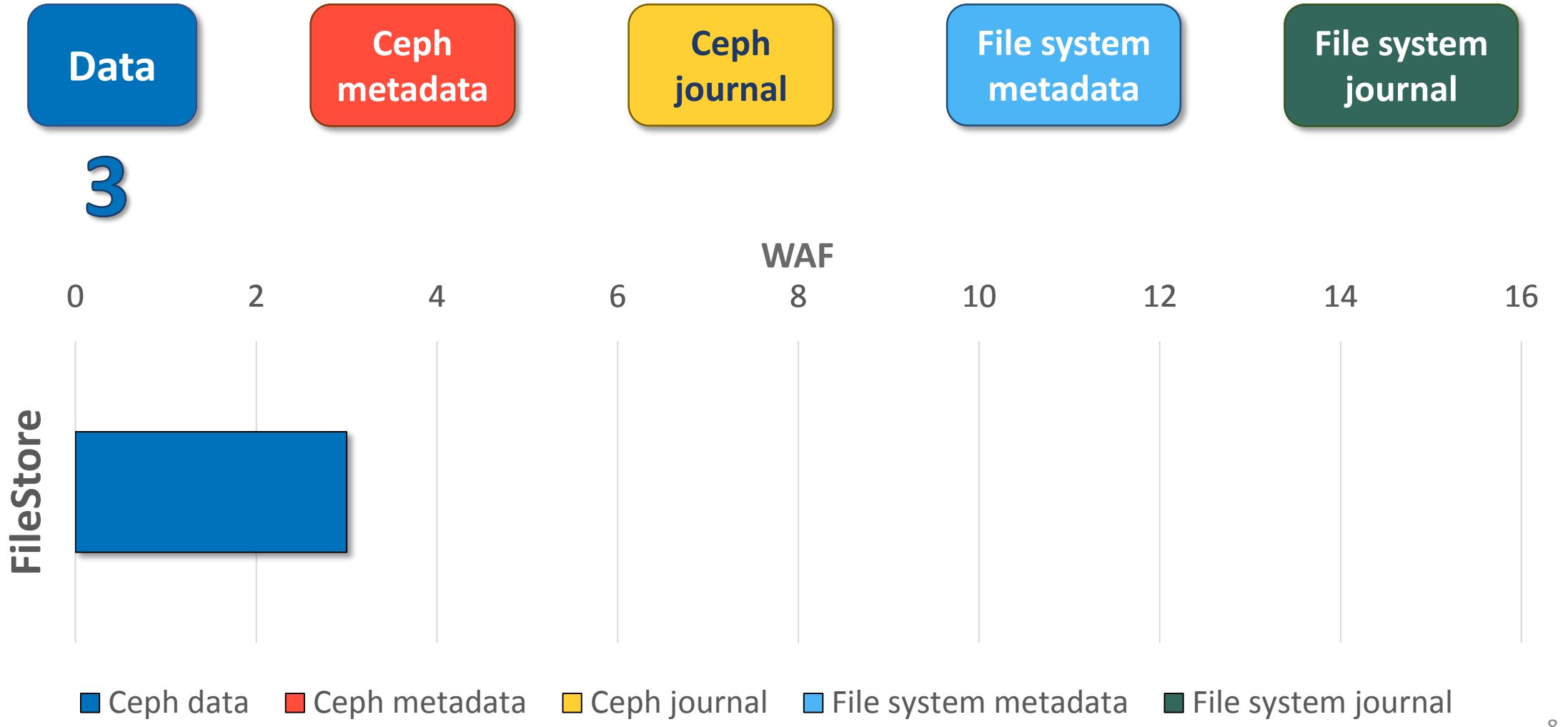
- Single write can make **several hidden I/Os**
- What about in long-term situation?



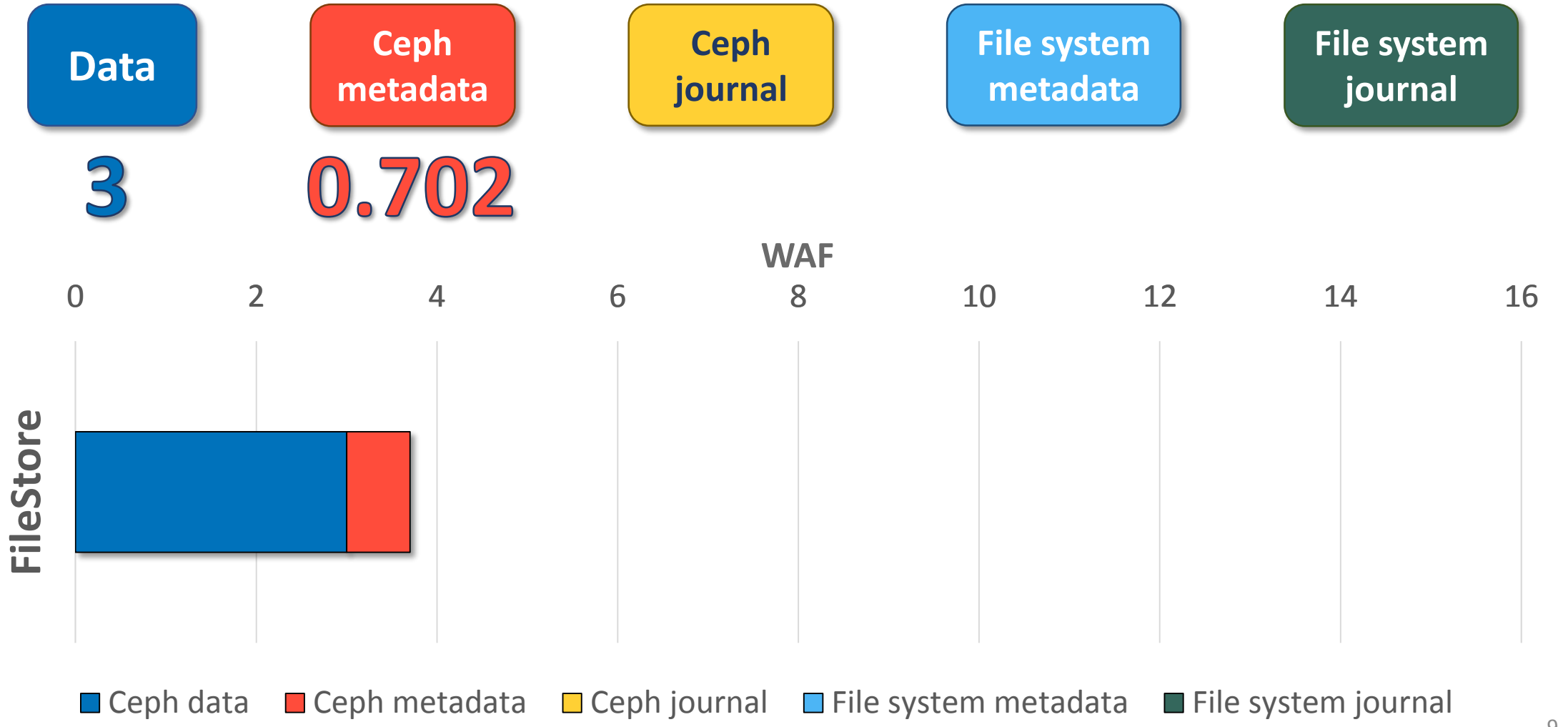
How Much Does Ceph Amplifies Writes



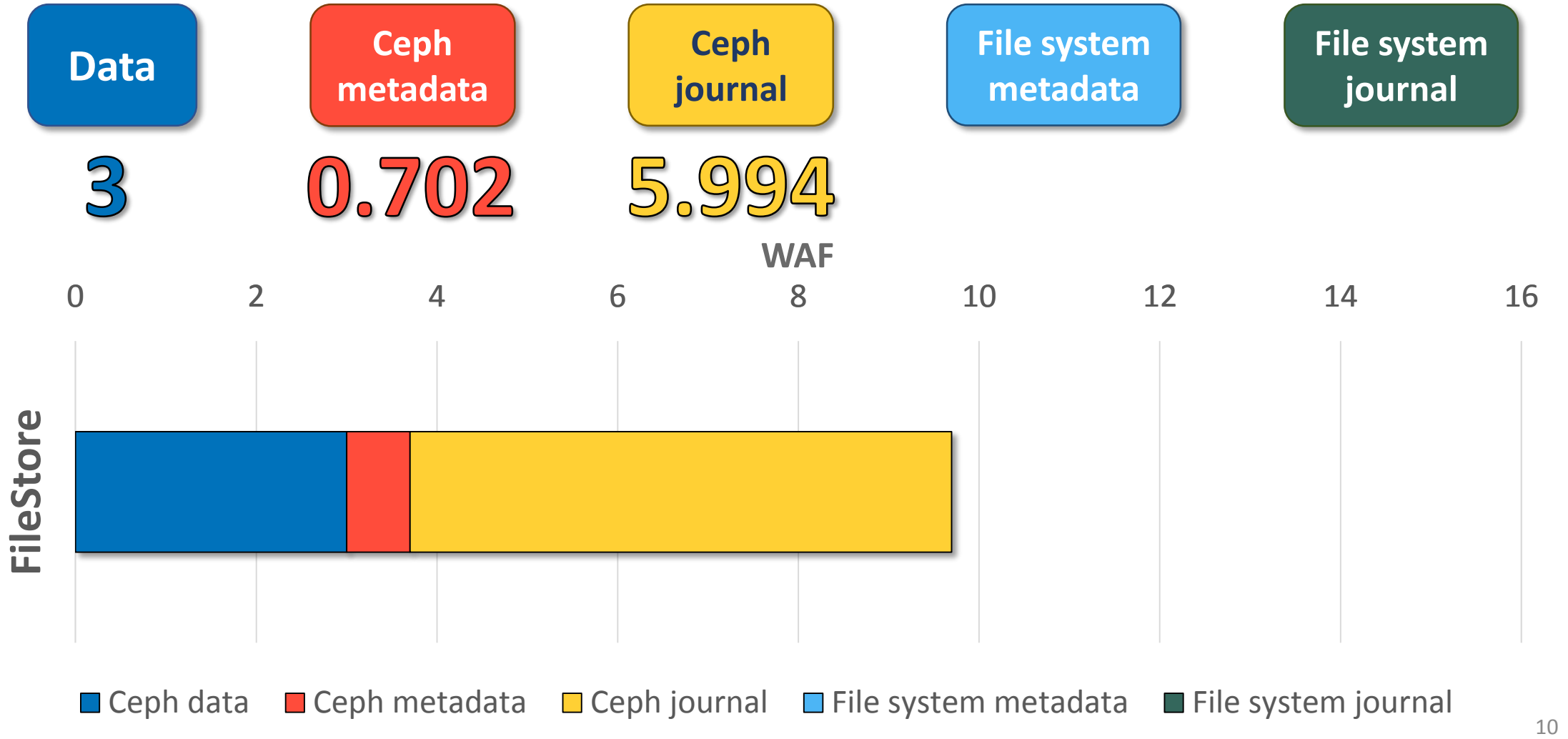
How Much Does Ceph Amplifies Writes



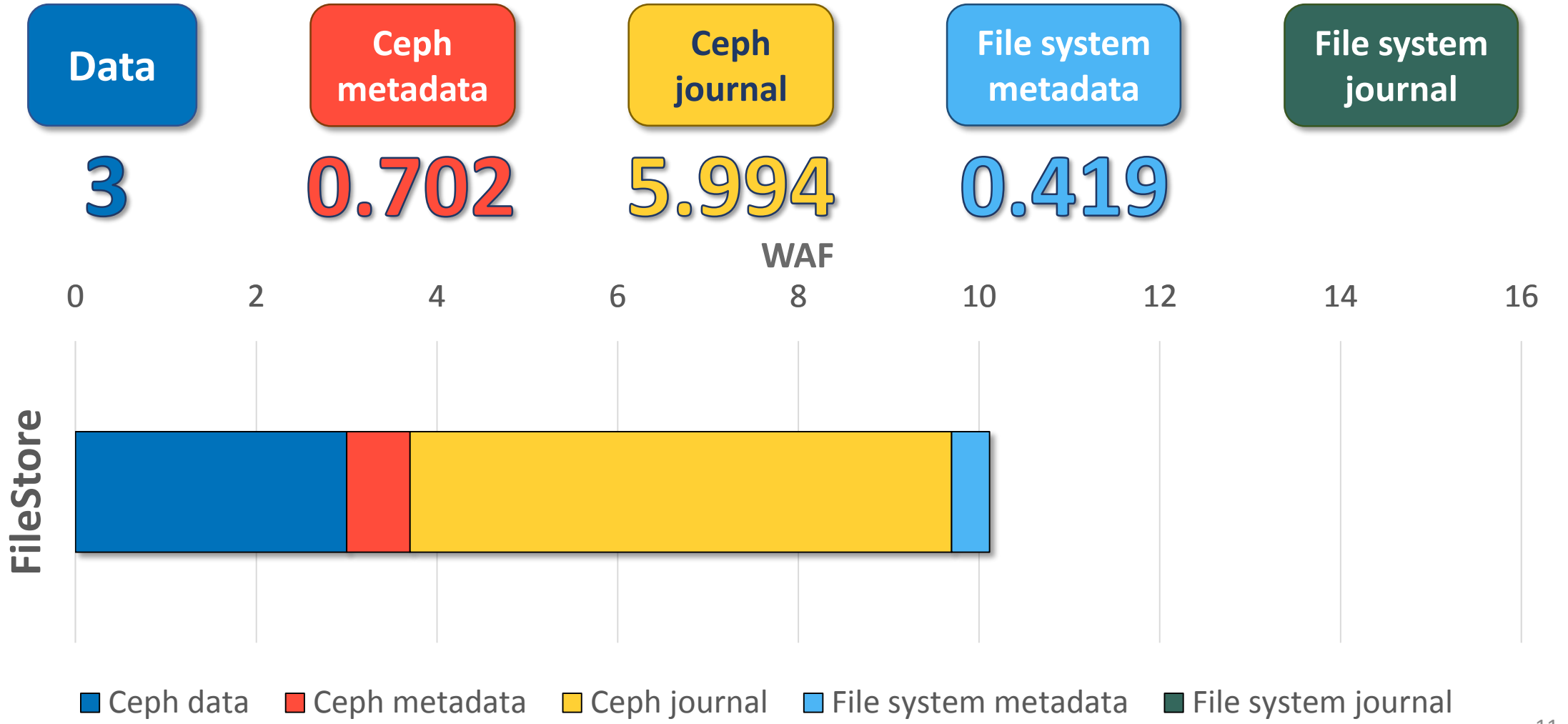
How Much Does Ceph Amplifies Writes



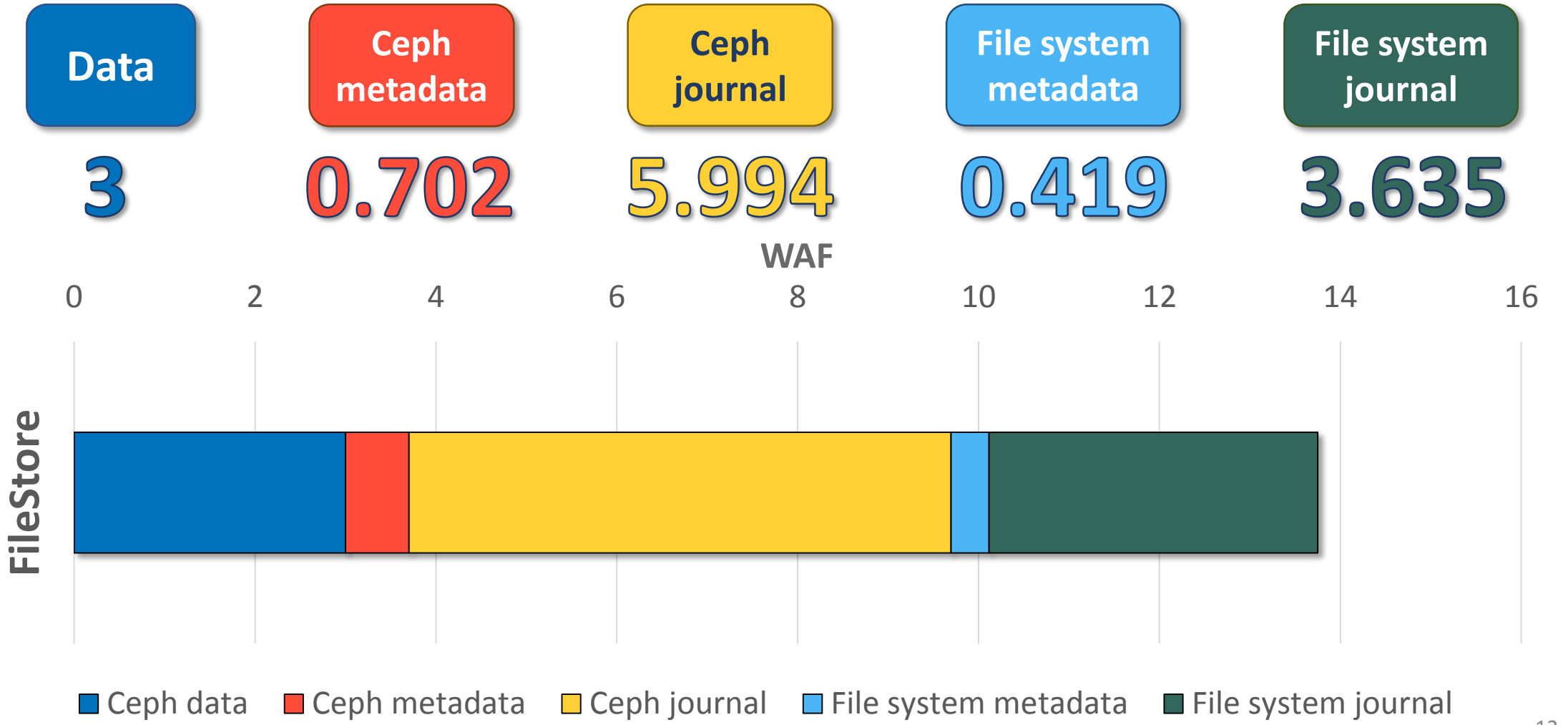
How Much Does Ceph Amplifies Writes



How Much Does Ceph Amplifies Writes



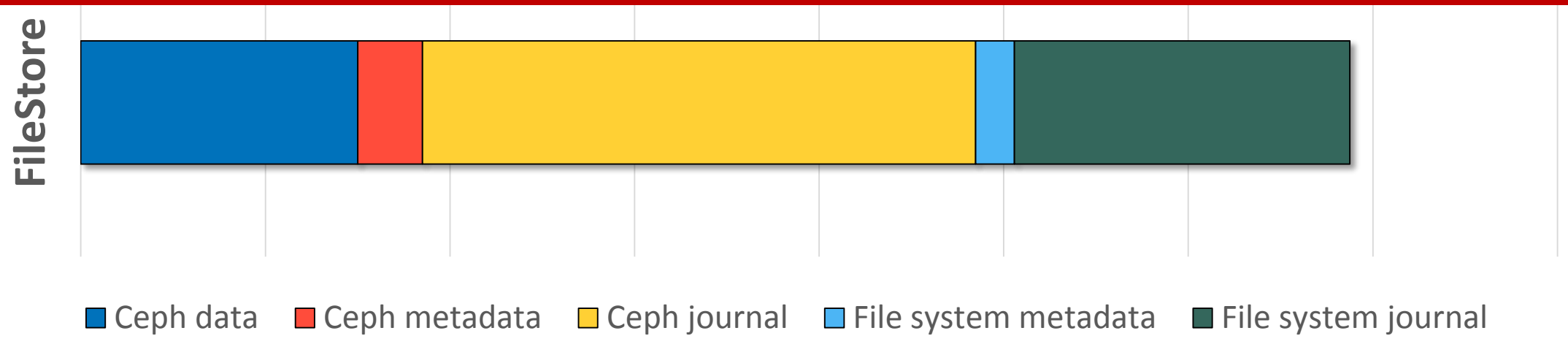
How Much Does Ceph Amplifies Writes



How Much Does Ceph Amplifies Writes

Data | Ceph metadata | Ceph journal | File system metadata | File system journal

Writes amplified by over **13x**



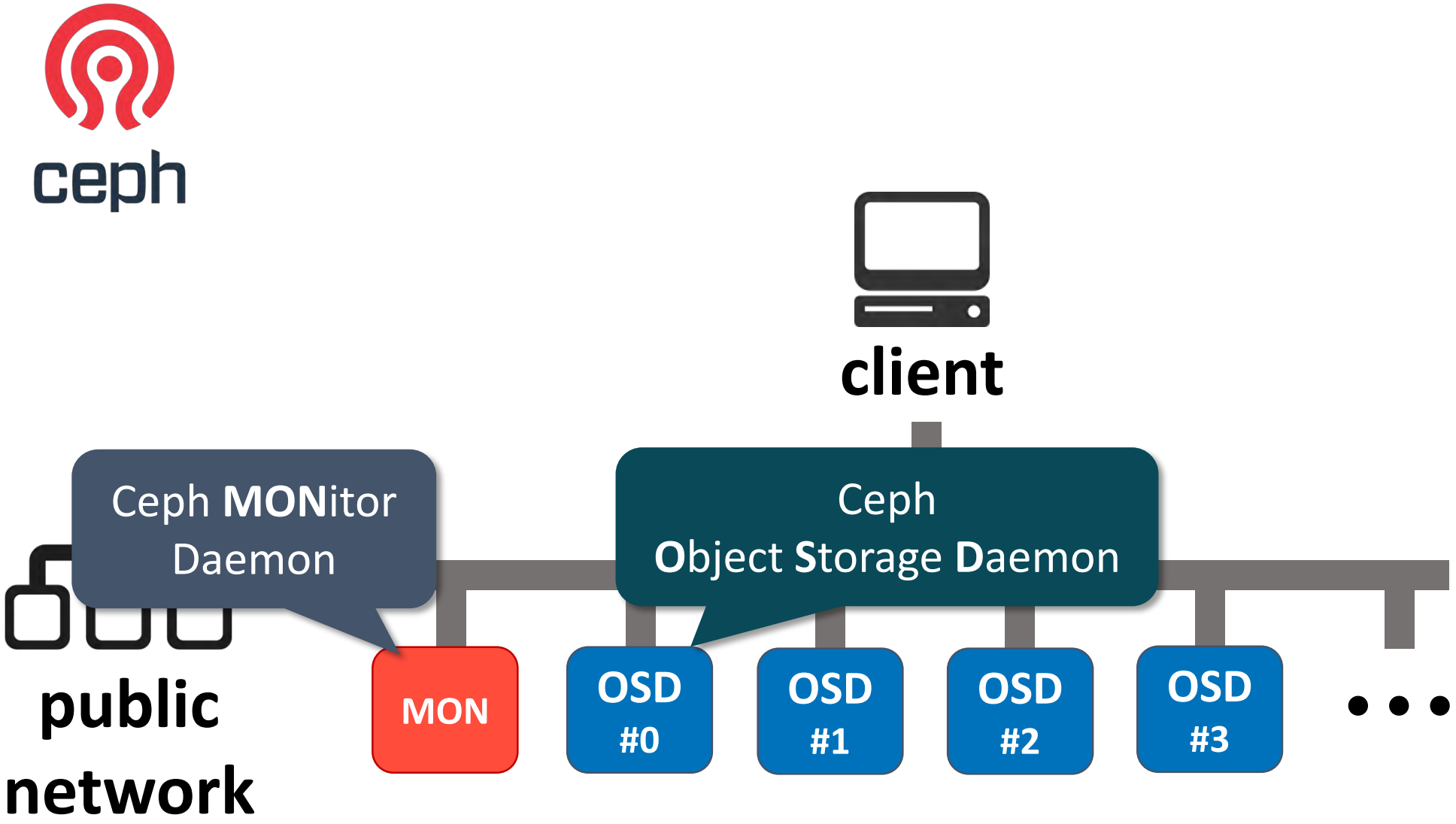
Our Motivation & Goal

- How/Why are writes highly amplified in Ceph?
  - With the exact numbers
- Why do we focus on write amplification?
  - WAF(Write amplification Factor) affects the overall performance
  - When using SSDs, it hurts the **lifespan** of SSDs
  - Larger WAF → smaller effective bandwidth
  - Redundant ***journaling of journal*** may exist
- Goals of this paper
  - **Understanding of write behaviors** of Ceph
  - **Empirical study** of write amplification in Ceph

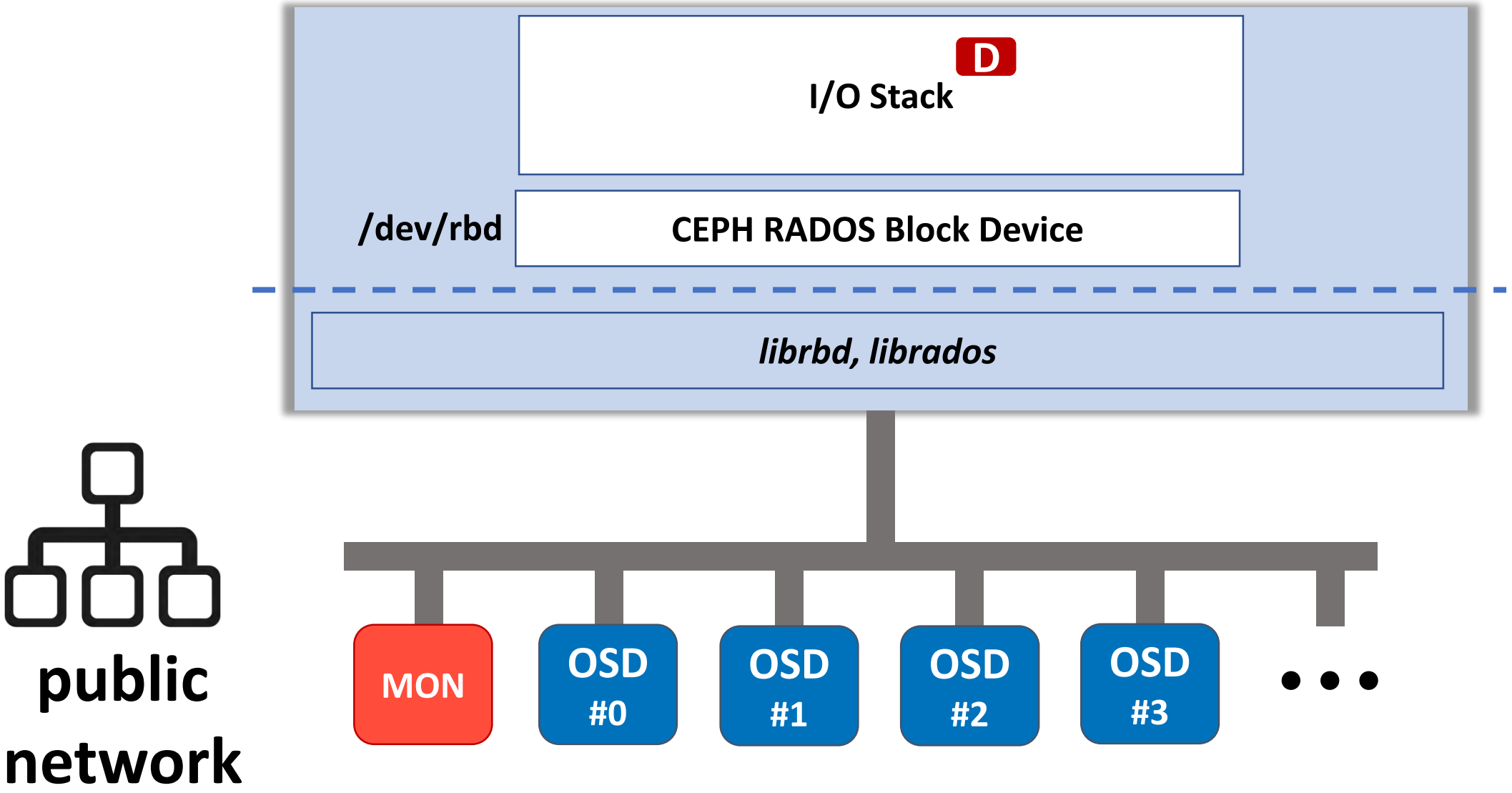
Outline

- Introduction
- **Background**
- Evaluation environment
- Result & Analysis
- Conclusion

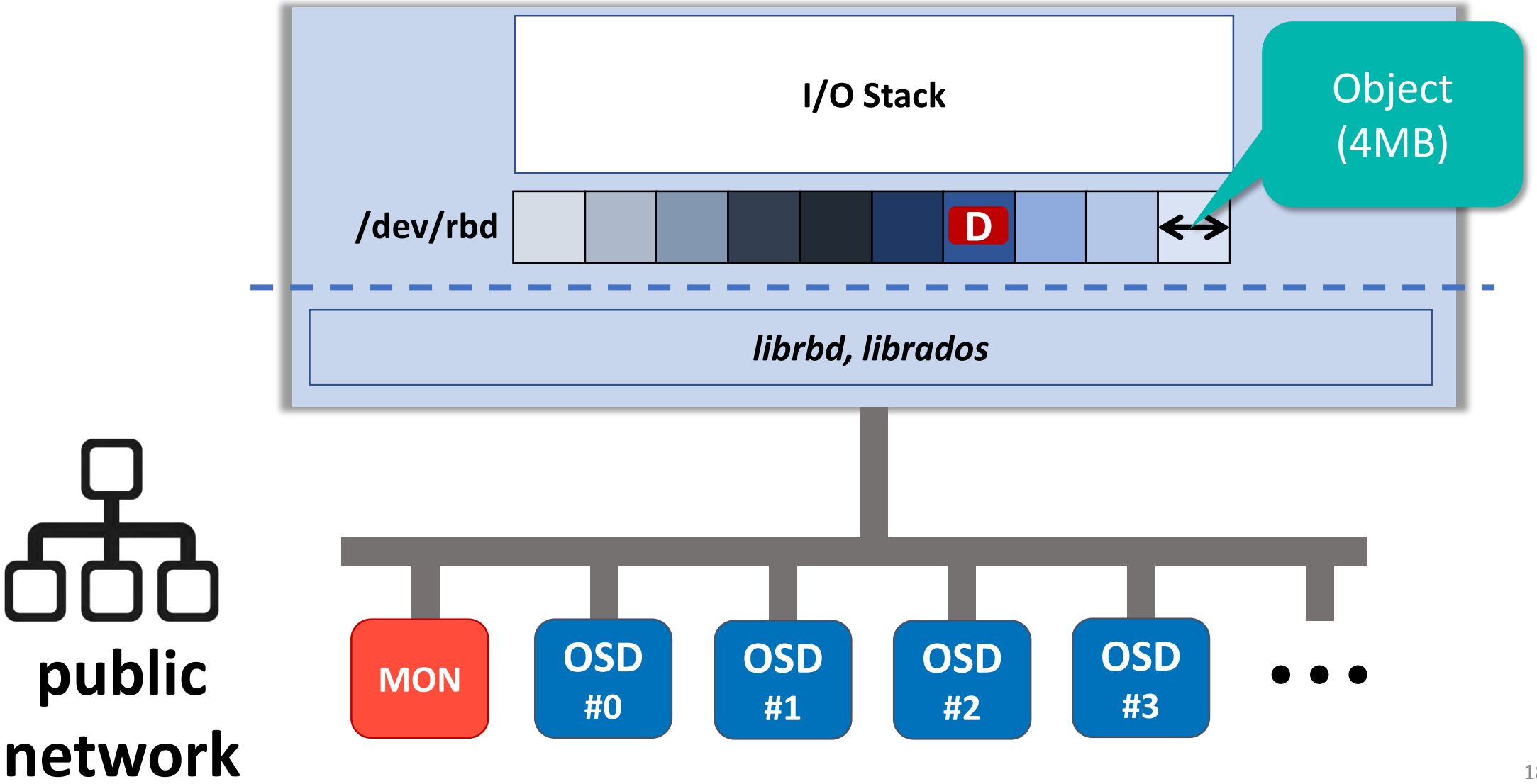
Architecture of Ceph RADOS Block Device (RBD)



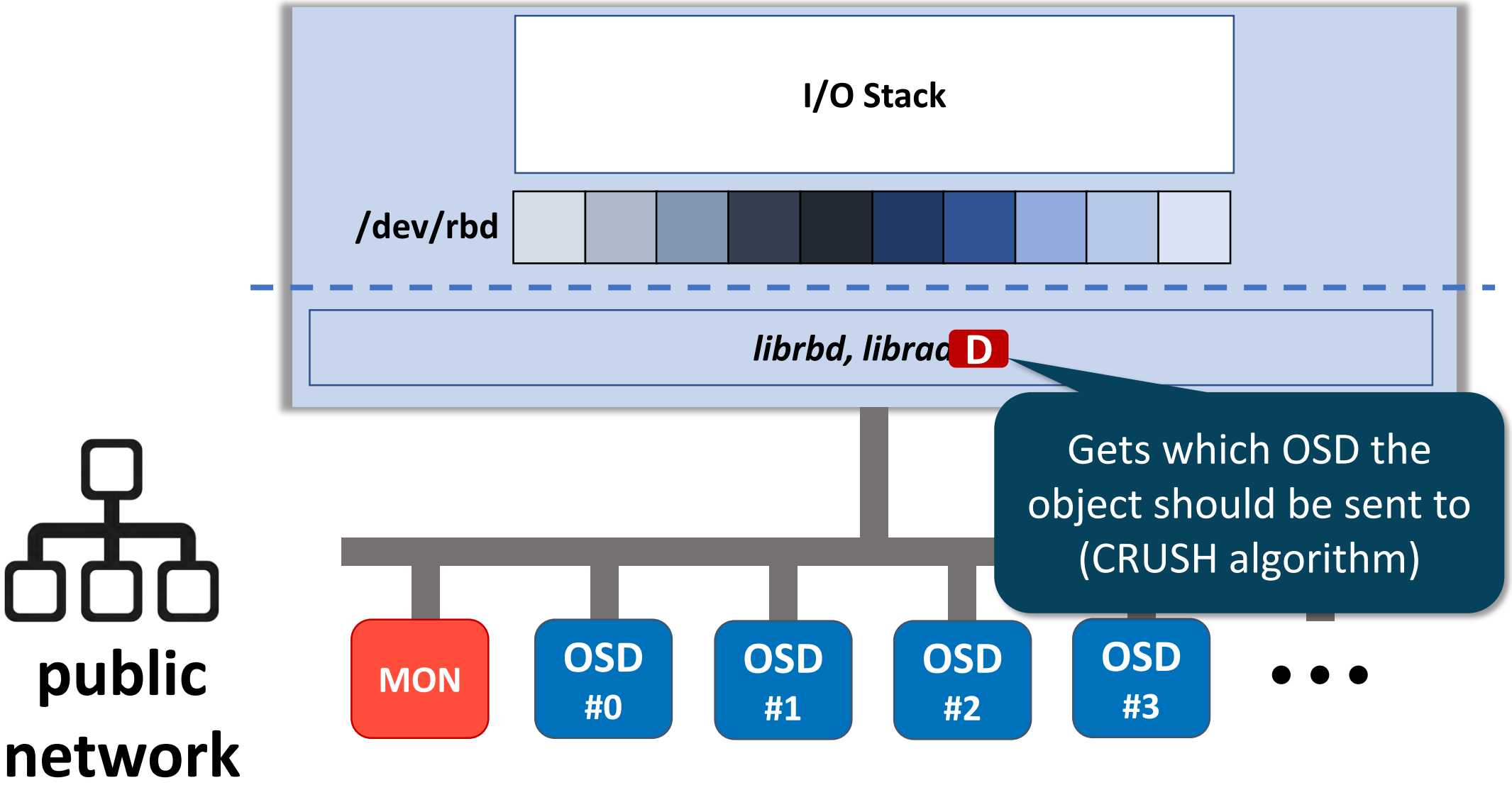
Architecture of Ceph RADOS Block Device (RBD)



Architecture of Ceph RADOS Block Device (RBD)



Architecture of Ceph RADOS Block Device (RBD)



Architecture of Ceph RADOS Block Device (RBD)

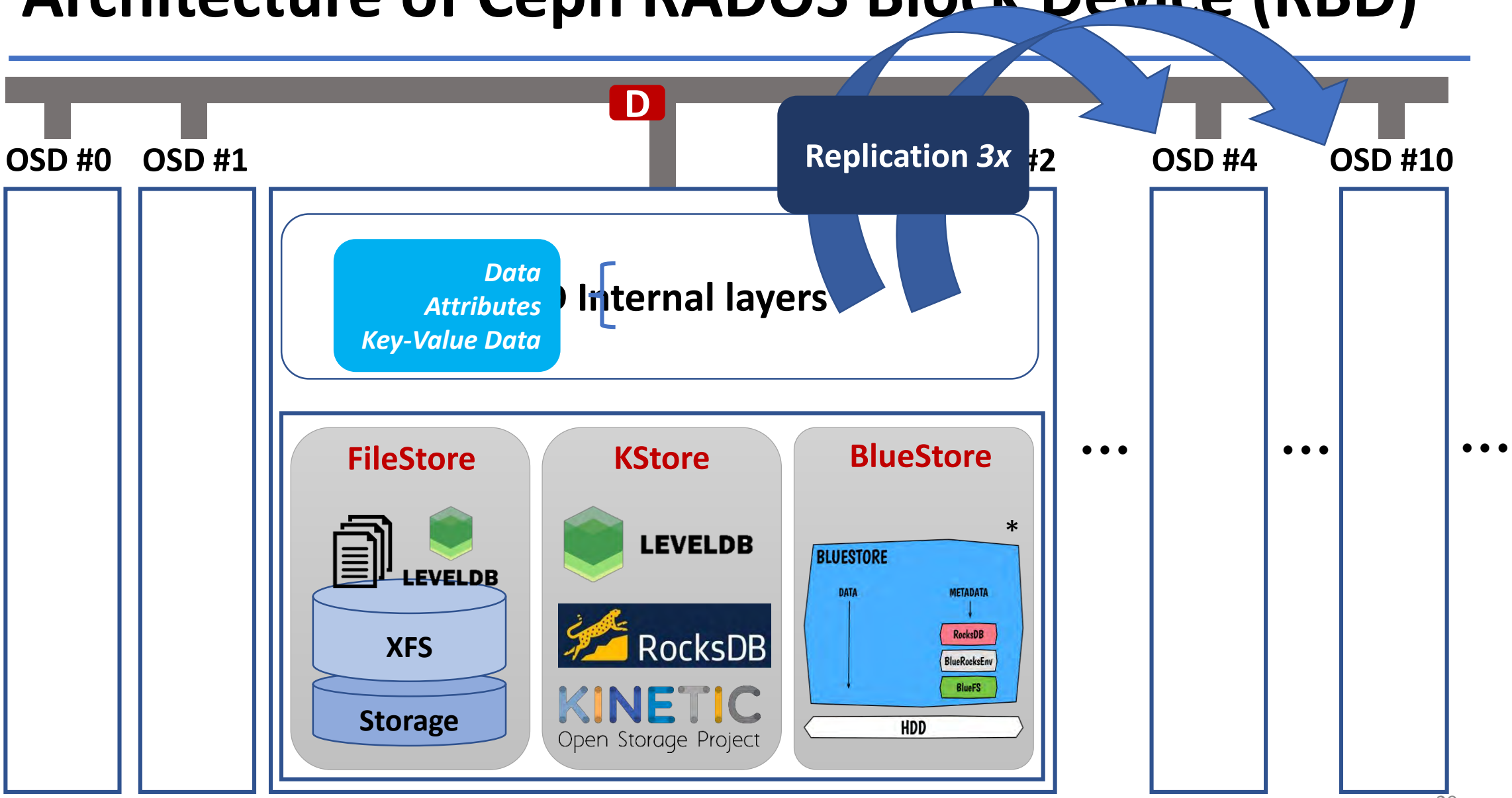


*Figure Courtesy: Sébastien Han https://www.sebastien-han.fr

Architecture of Ceph RADOS Block Device (RBD)

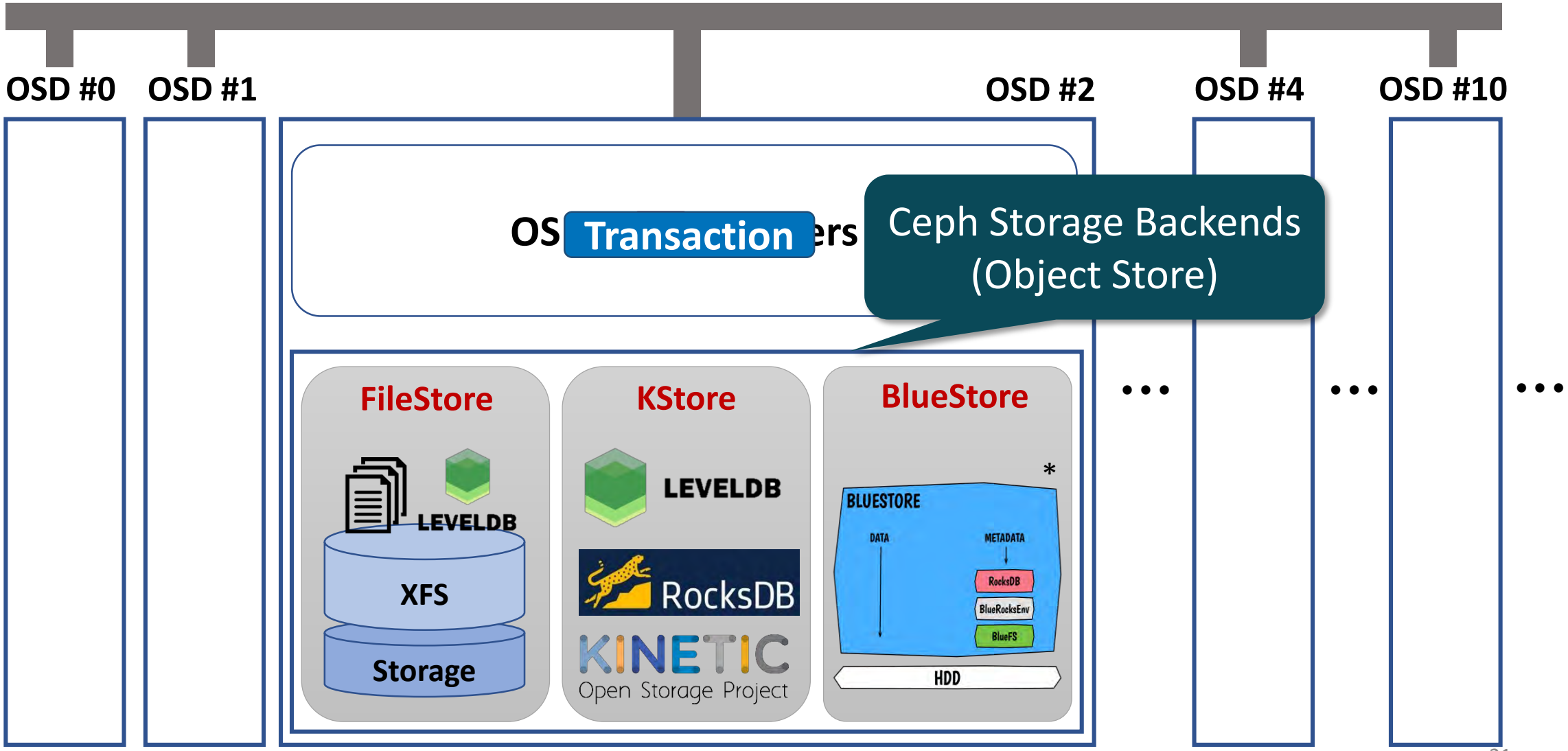
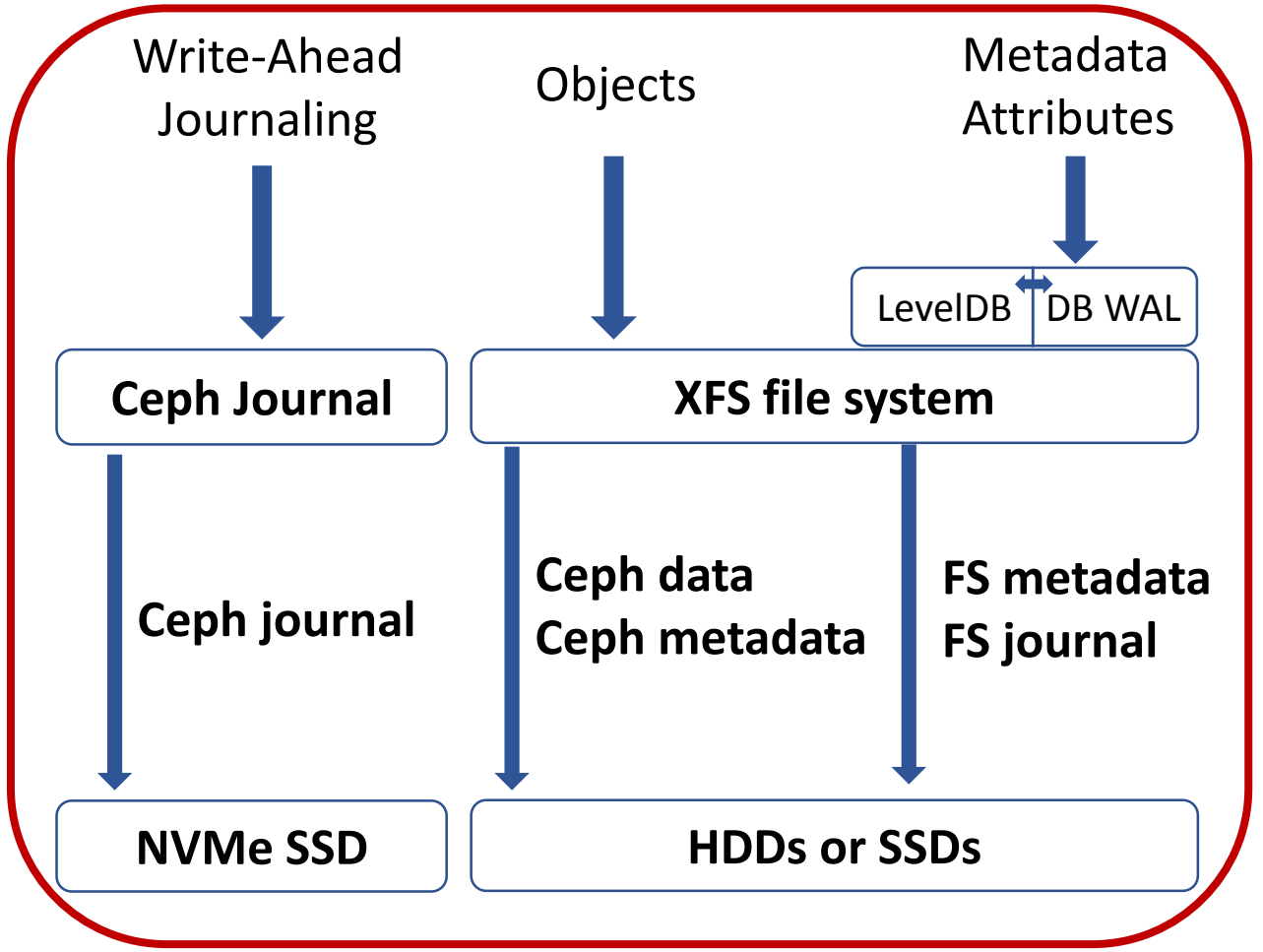


*Figure Courtesy: Sébastien Han https://www.sebastien-han.fr

Ceph Storage Backends: (1) FileStore

- Manages each object as a **file**

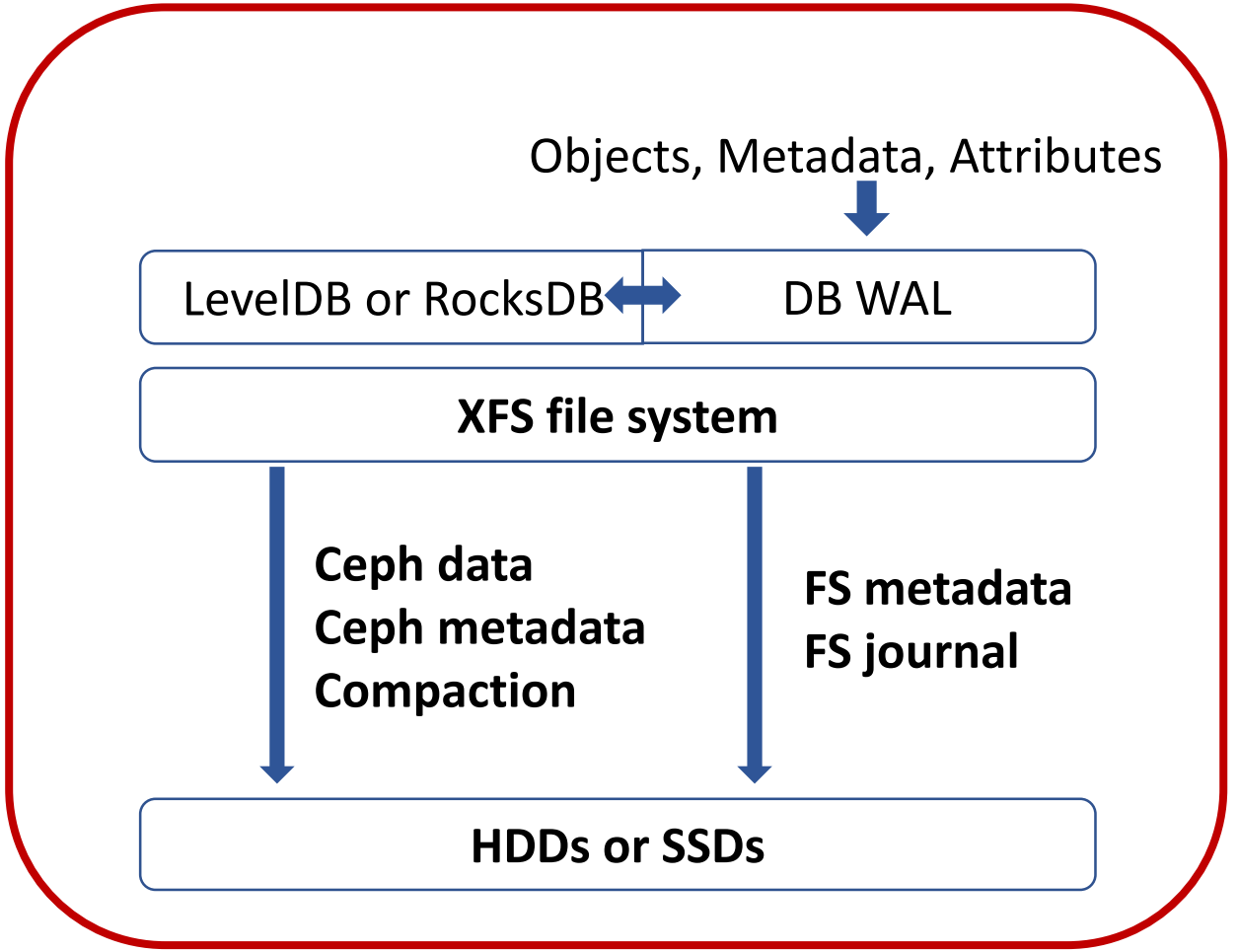
- Write flow in FileStore
  1. Write-Ahead journaling
     - For consistency and performance
  2. Performs actual write to file after journaling
     - Can be absorbed by page cache
  3. Calls *syncfs* + flush journal entries for every 5 seconds



<Breakdown of FileStore>

Ceph Storage Backends: (2) KStore

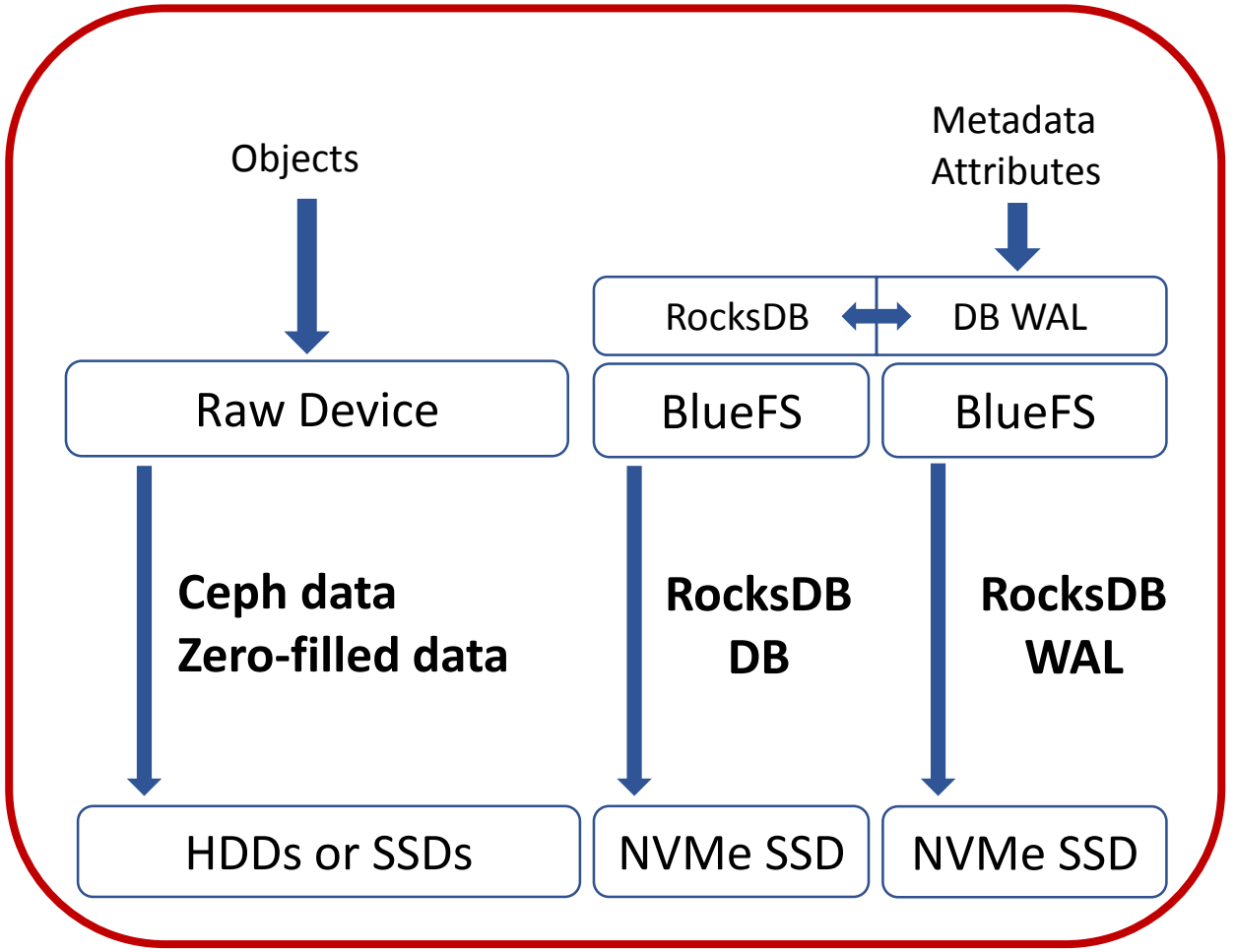
- Using existing **key-value stores**
  - Encapsulates everything to key-value pair
- Supports LevelDB, RocksDB and Kinetic Store
- Write flow in KStore
  1. Simply calls key-value APIs with the key-value pair



<Breakdown of KStore>

Ceph Storage Backends: (3) BlueStore

- To avoid limitations of FileStore
  - Double-write issue due to journaling
    - Directly stores data to raw device
- Write flow in BlueStore
  1. Puts data into raw block device
  2. Sets metadata to RocksDB on BlueFS (user-level file system)

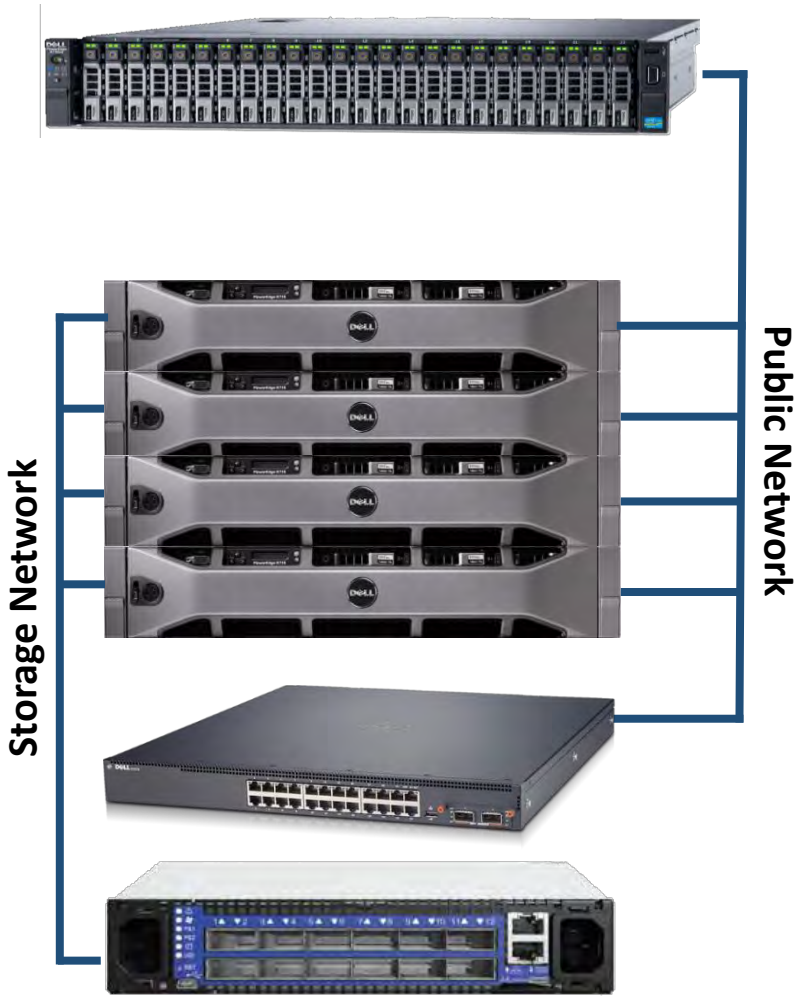


<Breakdown of BlueStore>

Outline

- Introduction
- Background
- **Evaluation environment**
- Result & Analysis
- Conclusion

H/W Evaluation Environment



Admin Server / Client (x1)	
Model	DELL R730XD
Processor	Intel® Xeon® CPU E5-2640 v3
Memory	128 GB

OSD Servers (x4)	
Model	DELL R730
Processor	Intel® Xeon® CPU E5-2640 v3
Memory	32 GB
Storage	HGST UCTSSC600 600 GB x4 Samsung PM1633 960 GB x4 Intel® 750 series 400 GB x2

Switch (x2)	
Public Network	DELL N4032 10Gbps Ethernet
Storage Network	Mellanox SX6012 40Gbps InfiniBand

S/W Evaluation Environment

- Linux 4.4.43 kernel
  - Some modifications to collect and classify write requests
- Ceph Jewel LTS version (v10.2.5)
- From client side
  - Configures a 64GiB KRBD
    - 4 OSDs per OSD server (total **16 OSDs**)
  - Generates workloads using *fio* while collecting trace and diskstats
- Perform 2 different workloads
  - Microbenchmark
  - Long-term workload

Outline

- Introduction
- Background
- Evaluation environment
- **Result & Analysis**
- Conclusion

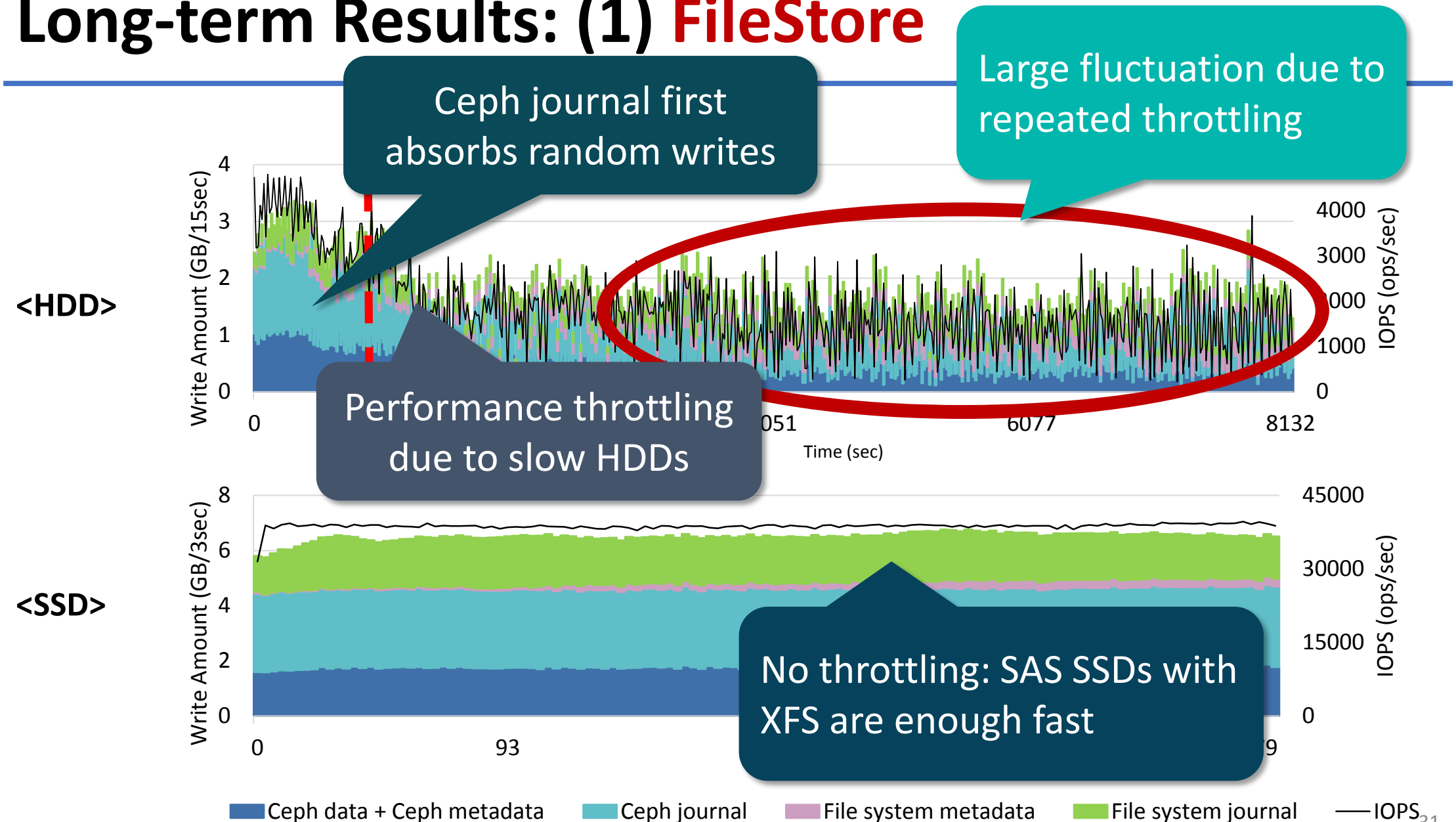
Key Findings in Microbenchmark

- FileStore
  - Large WAF when write request size is small (**over 40** at 4KB write)
  - WAF converges to 6 (3x by replication, 2x by Ceph journaling)
- KStore
  - Sudden WAF jumps due to *memtable* flush
  - WAF converges to 3 (by replication)
- BlueStore
  - Because the minimum extent size is set to 64KB
    - **Zero-filled data for the hole** within the object
    - **WAL** (Write-Ahead Logging) for small overwrite to guarantee durability
  - WAF converges to 3 (by replication)

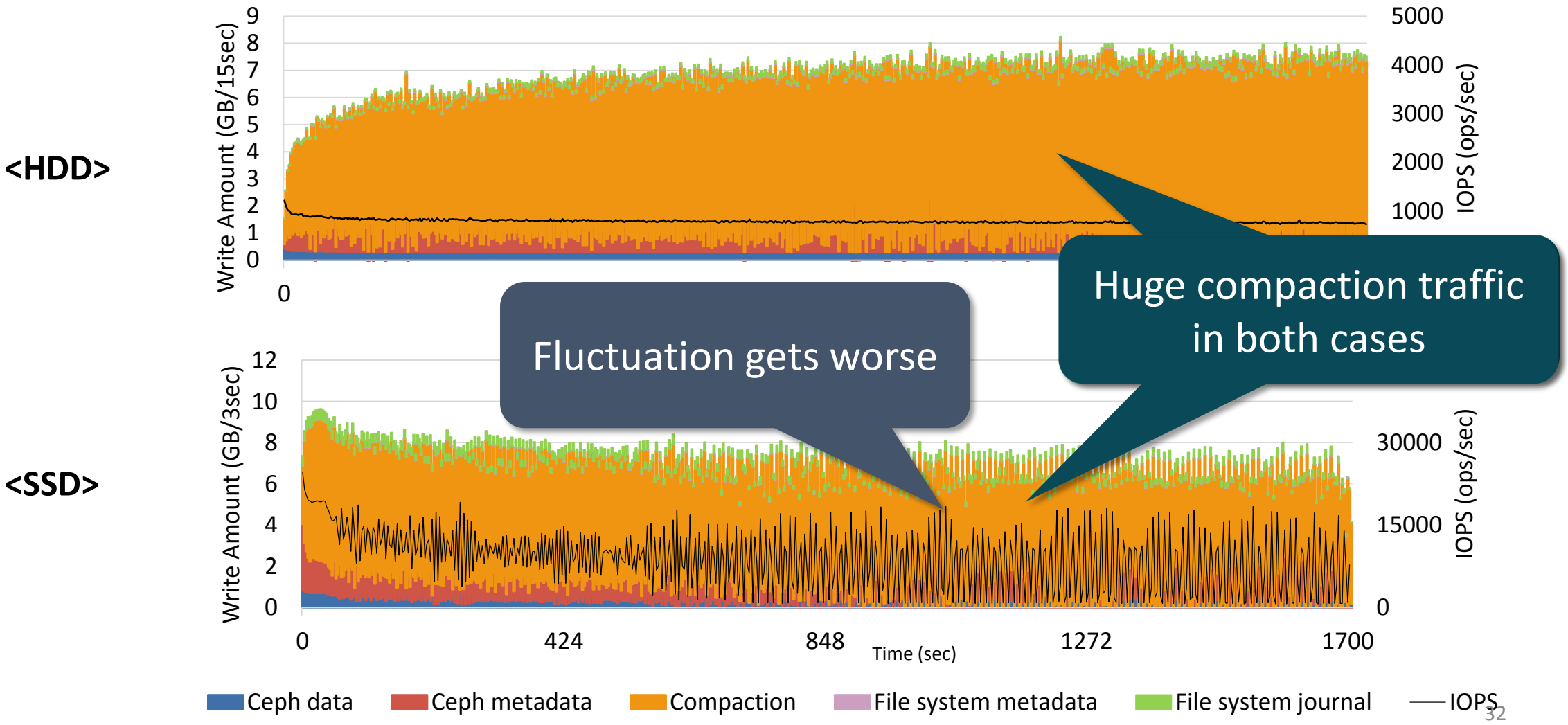
Long-Term Workload Methodology

- We focus on 4KB random writes
  - In VDI, most of the writes are **random** and **4KB in size**
- Workload scenario
  1. Install Ceph and create a krbd partition
  2. Drop page cache, call sync and wait for 600 secs
  3. Issue **4KB random writes** with QD=128 until the total write amount reaches 90% of the capacity (57.6GiB)
- Run tests with 16 HDDs first and repeat with 16 SSDs
- Calculate and **breakdown WAF** for given time period

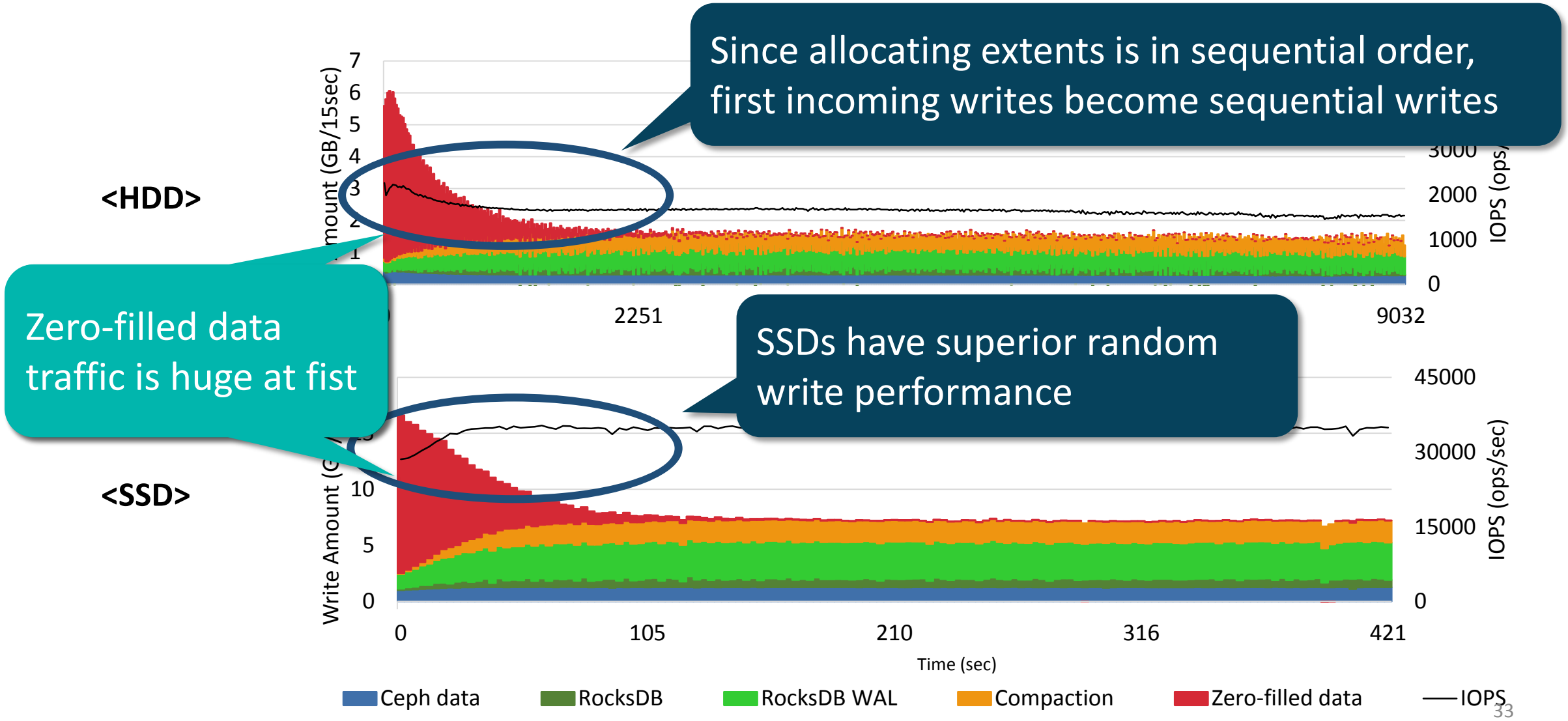
Long-term Results: (1) FileStore



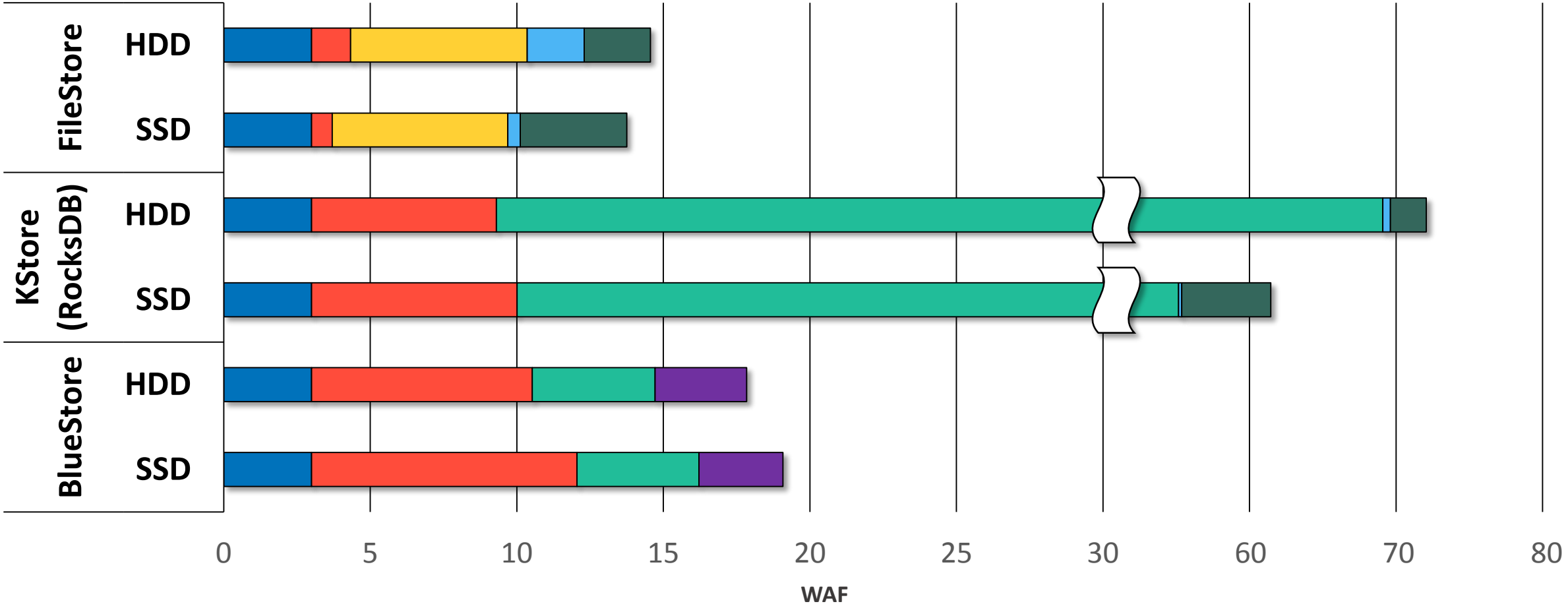
Long-term Results: (2) KStore (RocksDB)



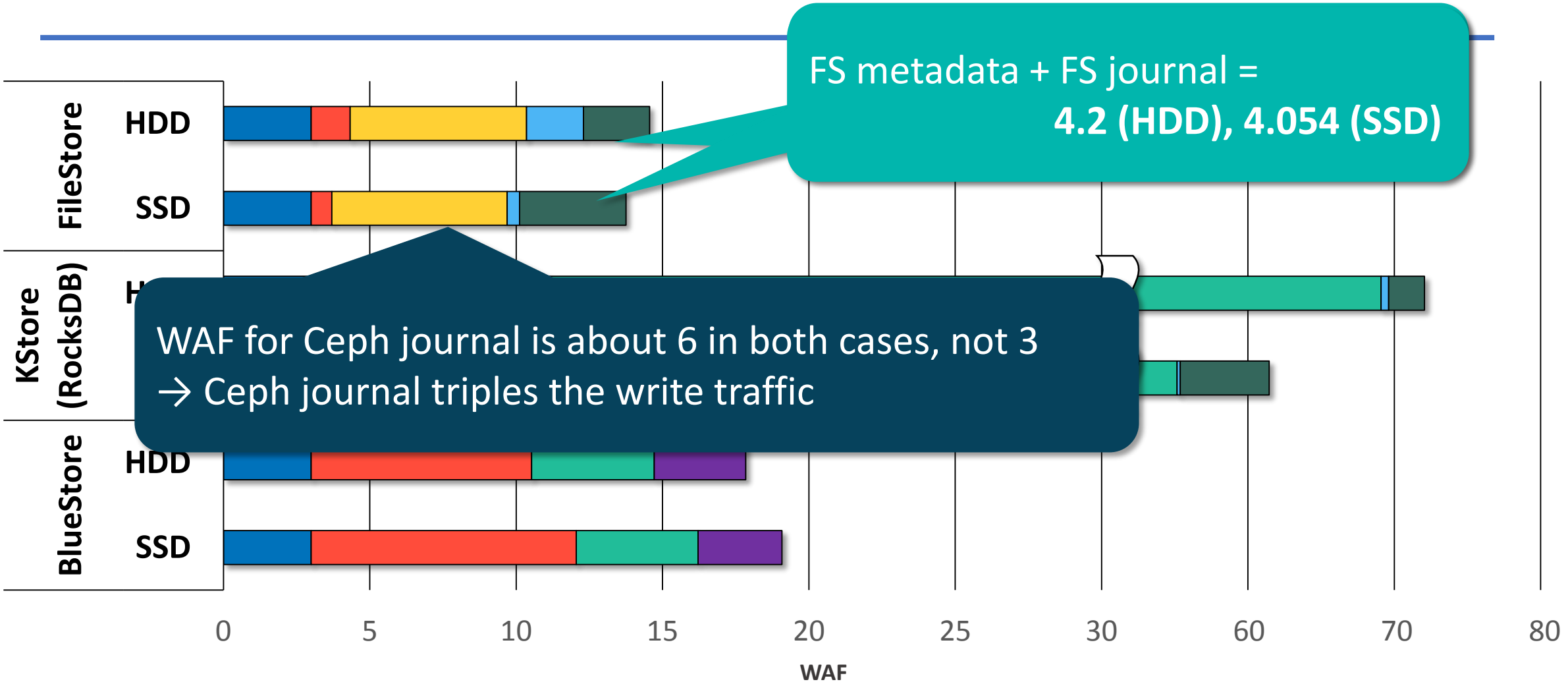
Long-term Results: (3) BlueStore



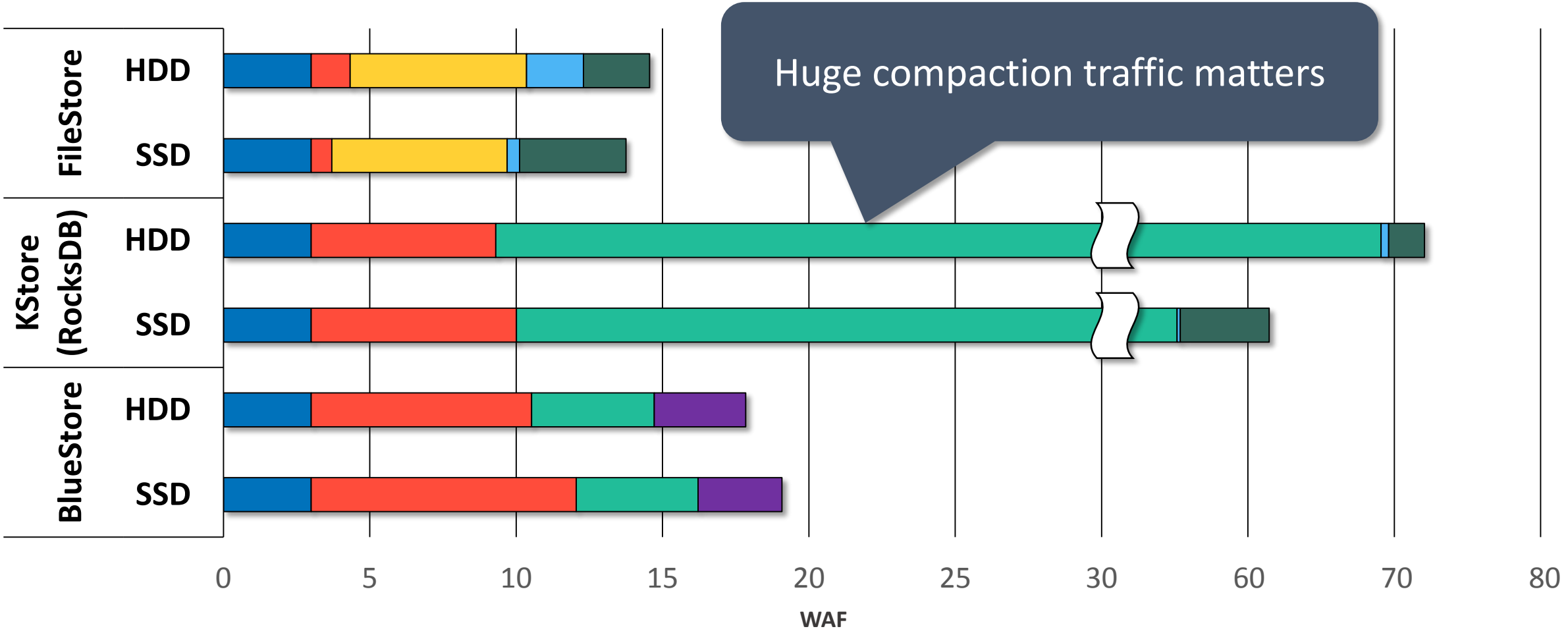
Overall Results of WAF



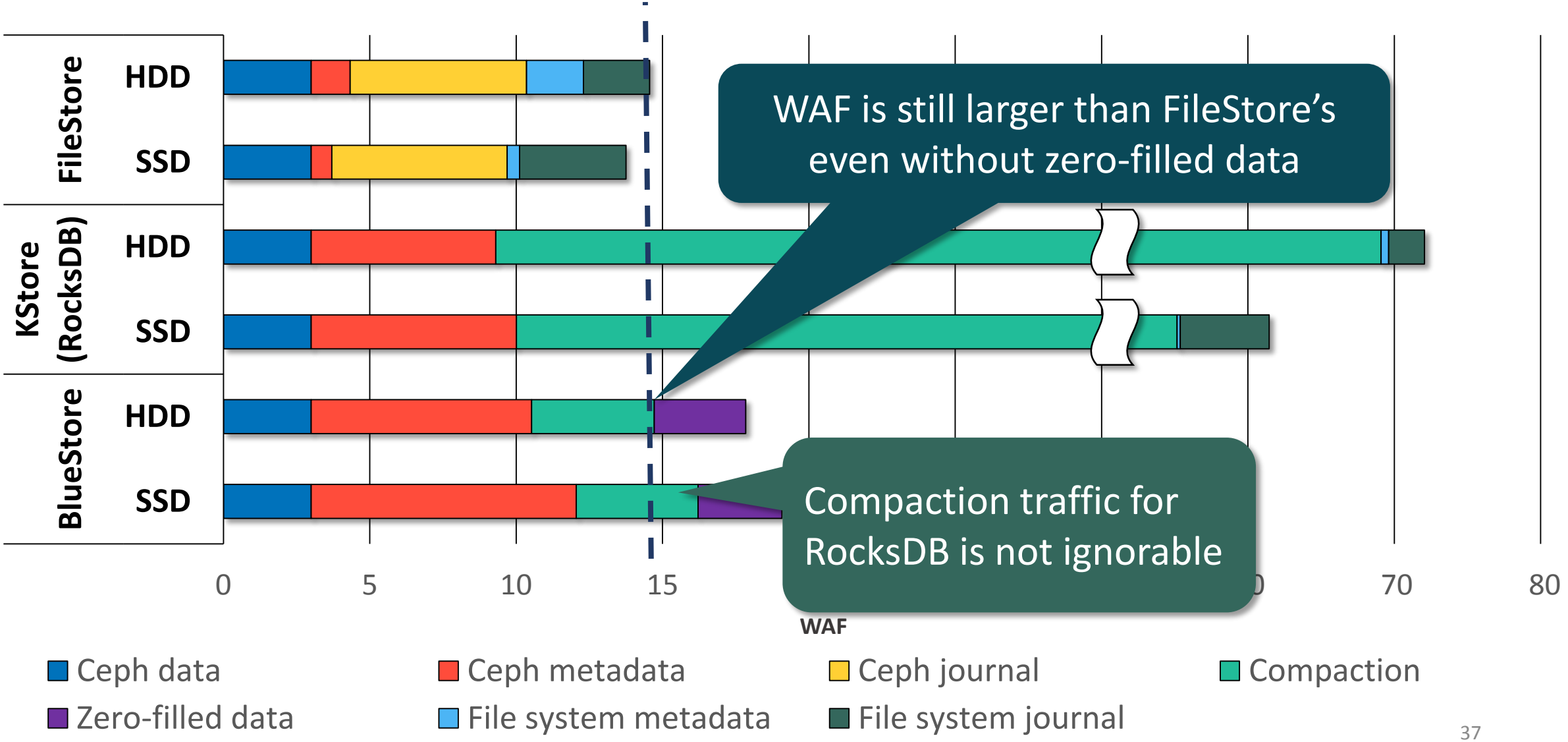
Overall Results of WAF



Overall Results of WAF



Overall Results of WAF



Conclusion

- Writes are amplified by more than 13x in Ceph
  - No matter which storage backend is used
- FileStore
  - External Ceph journaling triples write traffic
  - File system overhead exceeds the original data traffic
- KStore
  - Suffers huge compaction overhead
- BlueStore
  - Small write requests are logged on RocksDB WAL
  - Unignorable zero-filled data & compaction traffic

Thank you!