What's the Story in EBS Glory: Evolutions and Lessons in Building Cloud Block Store

Weidong Zhang, et. el.

Presented by Andrey Satarin (@asatarin)
May 2024

https://asatarin.github.io/talks/2024-05-evolution-of-cloud-block-store/

Outline

- EBS Evolution from EBS1 to EBS3 / EBSX
- Elasticity in latency and throughput
- Availability
- Conclusions and references

Cloud Block Store aka Elastic Block Store

Cloud Block Store

- Persistent Virtual Disk (VD) in the cloud
- Can attach to a virtual machine
- Can scale IOPS / throughput / capacity in a wide range

EBS Architecture Evolution

Timeline (EBS1+EBS2)

2012 — EBS1 (TCP / HDD)

2015 — EBS2 (Luna + RDMA / SSD)

  2016 — Background Erasure Coding / Compression

Timeline (EBS3 + EBSX)

2019 — EBS3 (Solar + RDMA / SSD)

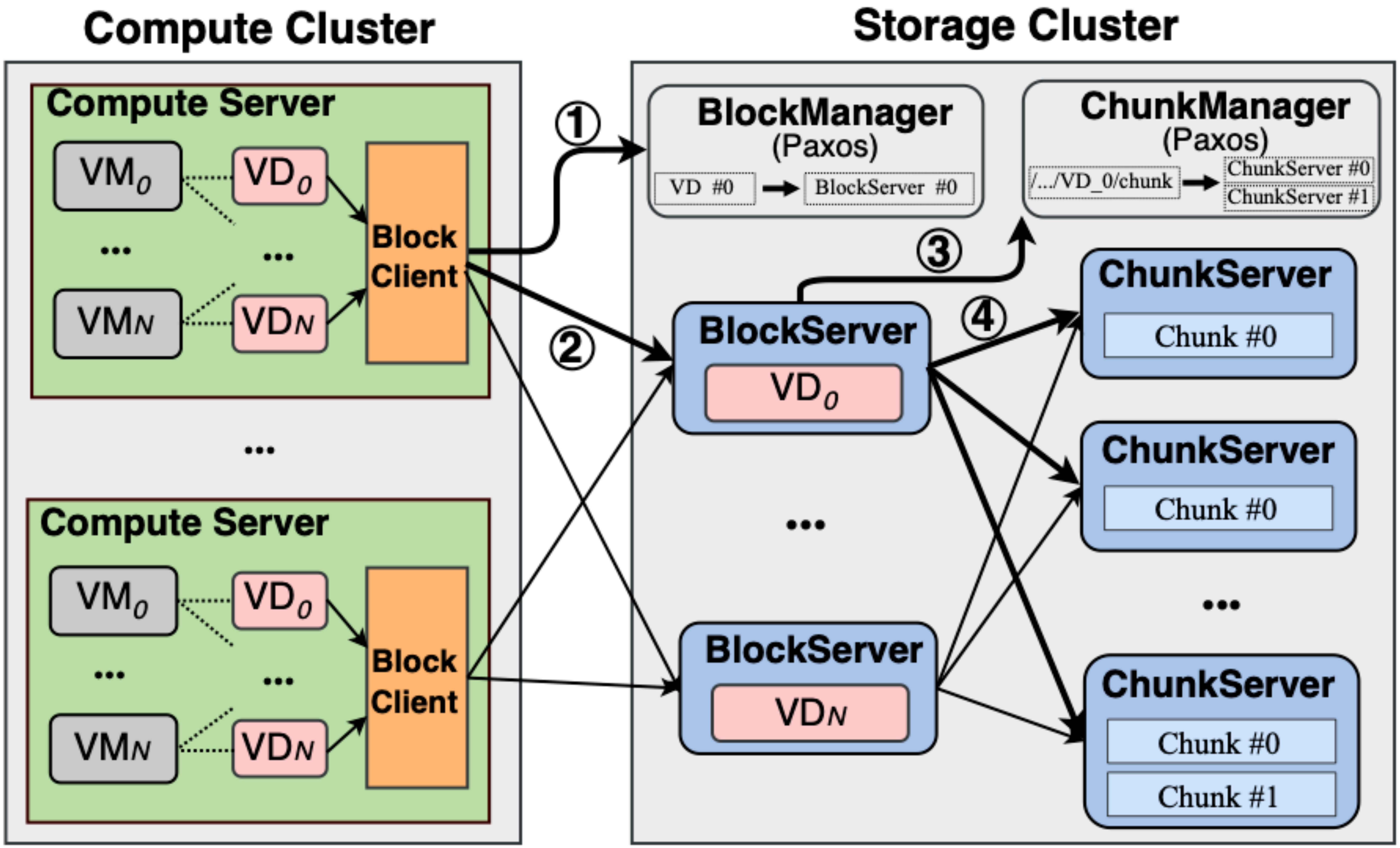
- 2020 — **Foreground Erasure Coding / Compression**
- 2021 — AutoPerformanceLevel (AutoPL)
- 2021 — **Logical Failure Domain**

2022 — EBSX (One Hop Solar / PMem + SSD)

- 2024 — **Federated Block Manager**

EBS1: An Initial Foray

EBS1



Compute Cluster
Storage Cluster
Compute Server
VM0
VD0
...
VMN
VDN
Block Client
Compute Server
VM0
VD0
VMN
VDN
Block Client
BlockManager
(Paxos)
VD #0
BlockServer #0
ChunkManager
(Paxos)
/.../VD_0/chunk
ChunkServer #0
ChunkServer #1
BlockServer
VD0
BlockServer
VDN
ChunkServer
Chunk #0
ChunkServer
Chunk #0
ChunkServer
Chunk #0
Chunk #1

EBS1 Architecture

- BlockManager (Paxos) maintains metadata about Virtual Disk (VD)
- BlockClient caches VD to block mappings
- Data abstraction of chunk — 64Mb of data
- ChunkManager (Paxos) stores metadata about chunks
- 3 way replicated on top of local Ext4 file system
- In-place updates

EBS1 Limitations

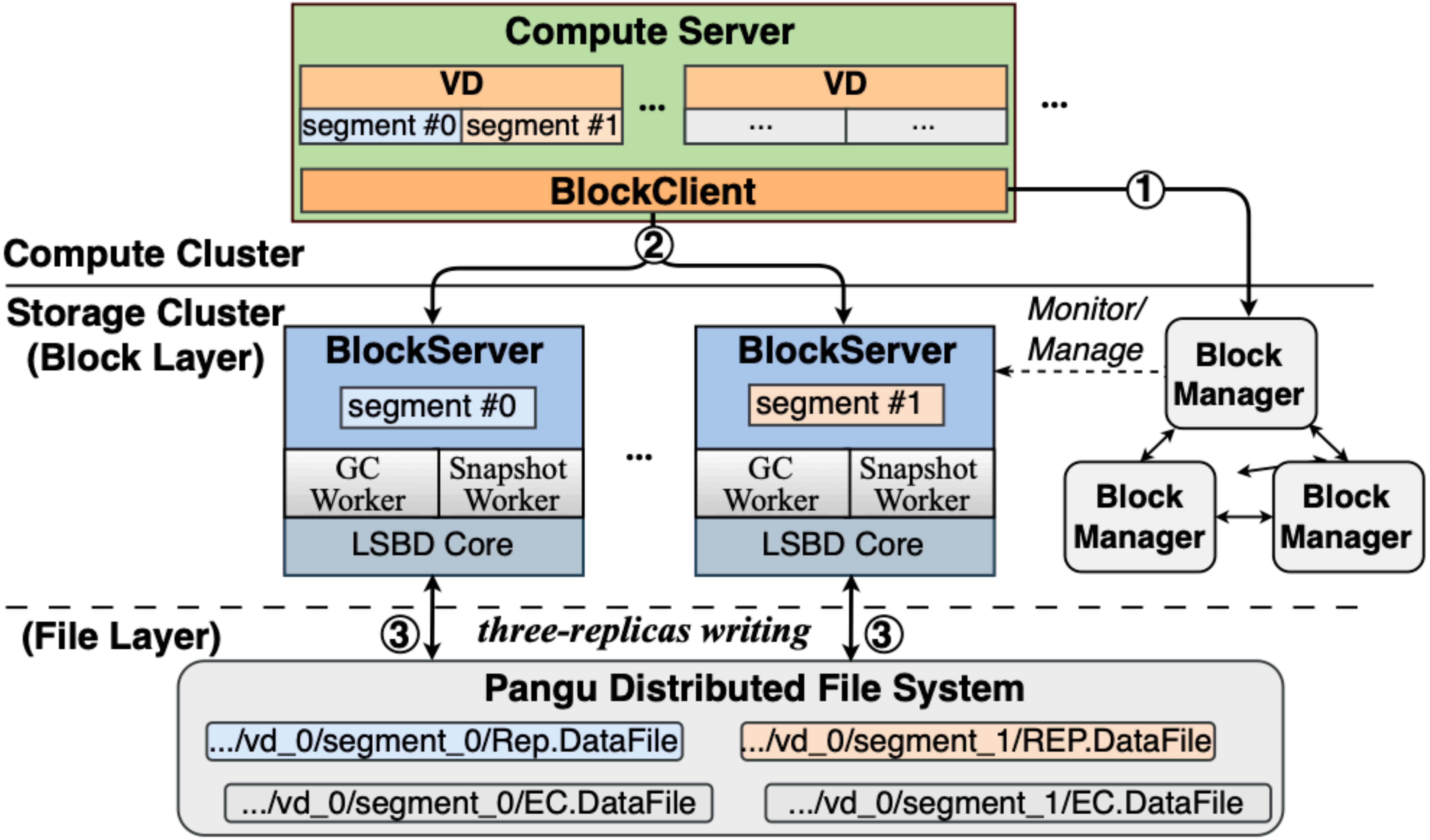
- **3x space overhead** due to replication
- **Limits** in performance and efficiency
- VD performance is **bound by a single BlockServer** performance
    - Might suffer from hotspots
- Hard to quantify and guarantee **SLO** with HDD and kernel TCP/IP

EBS2: Speedup with Space Efficiency

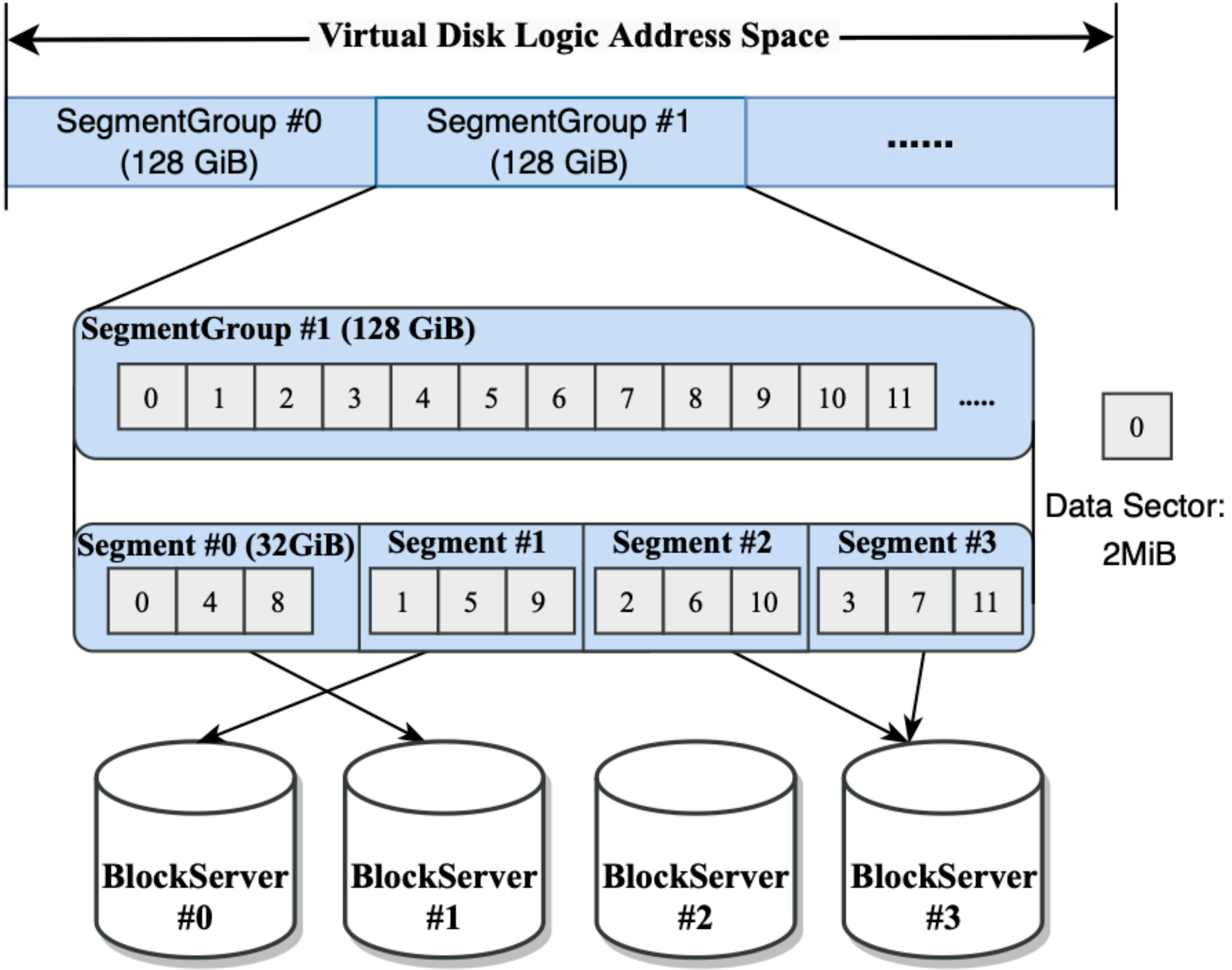
EBS2 Overview

- Does not directly handle persistence or consensus
- Built on top of distributed file system **Pangu**
- **Log-Structured** design of BlockServers translates writes into appends
- Traffic split into **frontend** (client I/O) and **backend** (GC, compression)
- Failover at the **granularity of a segment** instead of VD

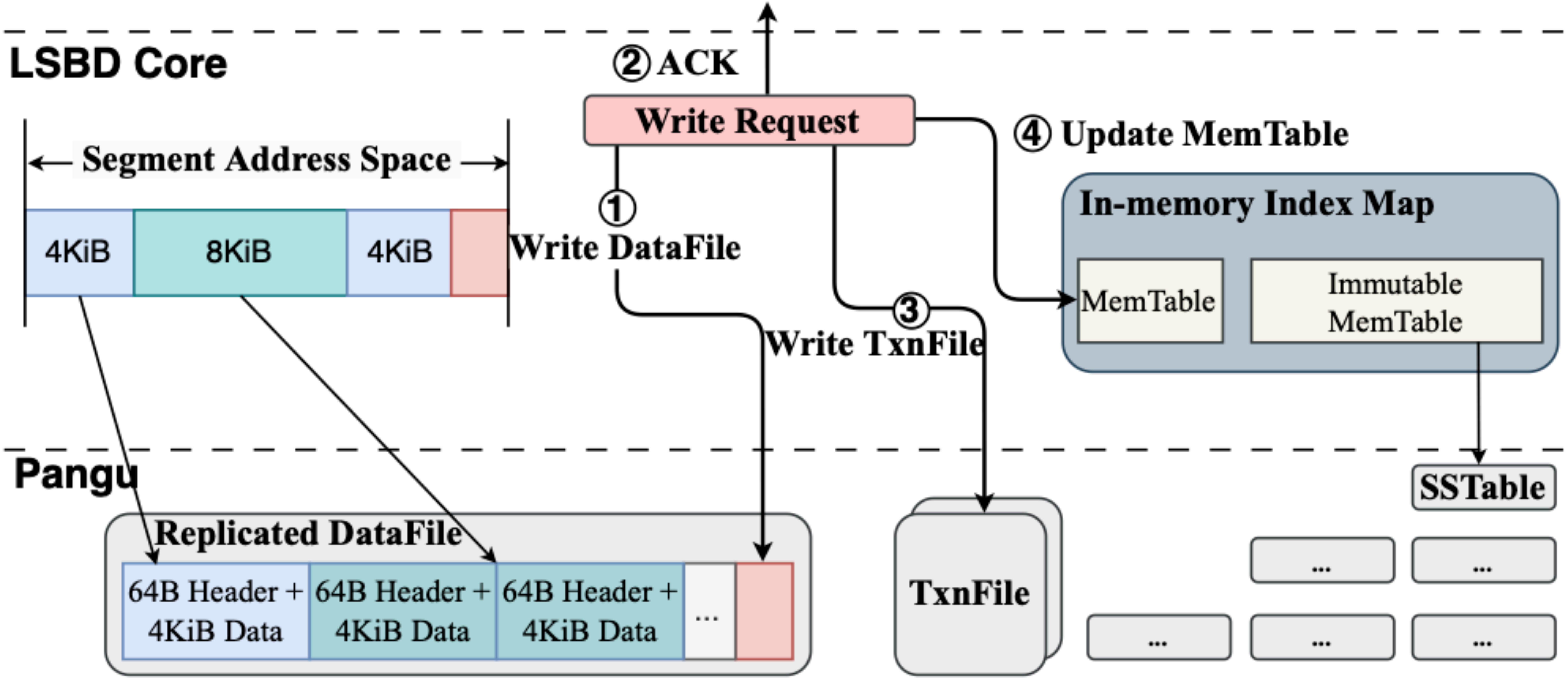
EBS2


Compute Server
VD
segment #0
segment #1
...
VD
...
...
...
BlockClient
1
2
Compute Cluster
Storage Cluster
(Block Layer)
BlockServer
segment #0
GC Worker
Snapshot Worker
LSBD Core
...
BlockServer
segment #1
GC Worker
Snapshot Worker
LSBD Core
Monitor/ Manage
Block Manager
Block Manager
Block Manager
(File Layer)
3
three-replicas writing
3
Pangu Distributed File System
.../vd_0/segment_0/Rep.DataFile
.../vd_0/segment_1/REP.DataFile
.../vd_0/segment_0/EC.DataFile
.../vd_0/segment_1/EC.DataFile

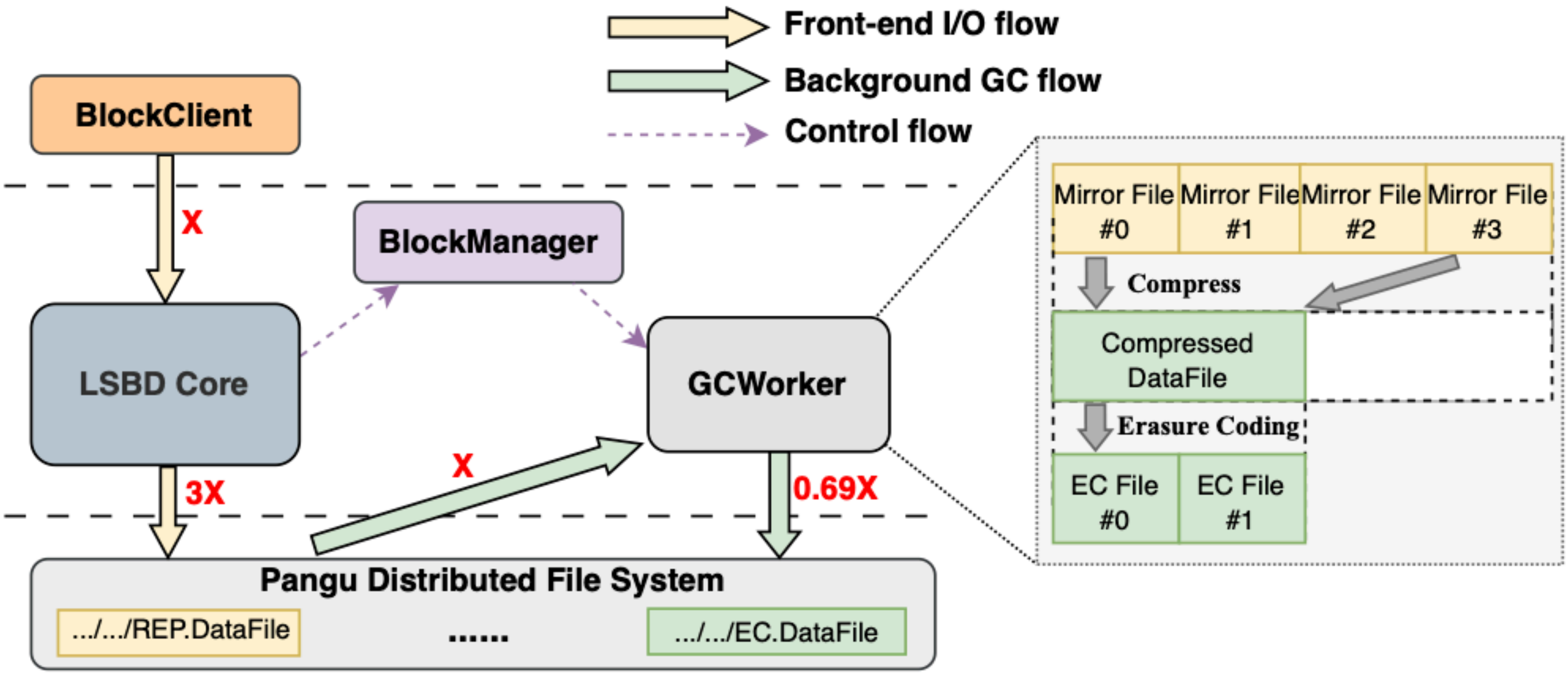
Disk


Virtual Disk Logic Address Space
SegmentGroup #0 (128 GiB)
SegmentGroup #1 (128 GiB)
......
SegmentGroup #1 (128 GiB)
0 1 2 3 4 5 6 7 8 9 10 11 .....
Segment #0 (32GiB)
0 4 8
Segment #1
1 5 9
Segment #2
2 6 10
Segment #3
3 7 11
0
Data Sector: 2MiB
BlockServer #0
BlockServer #1
BlockServer #2
BlockServer #3

Log-Structured Block Device (LSBD)



Garbage Collection



EBS2 by the Numbers

- Max IOPS 1M (10**6) — **50x** compared to EBS1
- Max throughput 4000 MiB/s — **13x** compared to EBS1
- Heavy **network amplification of 4.69x** — compared to 3x in EBS1
- Average space amplification of **1.29x** — compared to 3x in EBS1

EBS2 by the Numbers

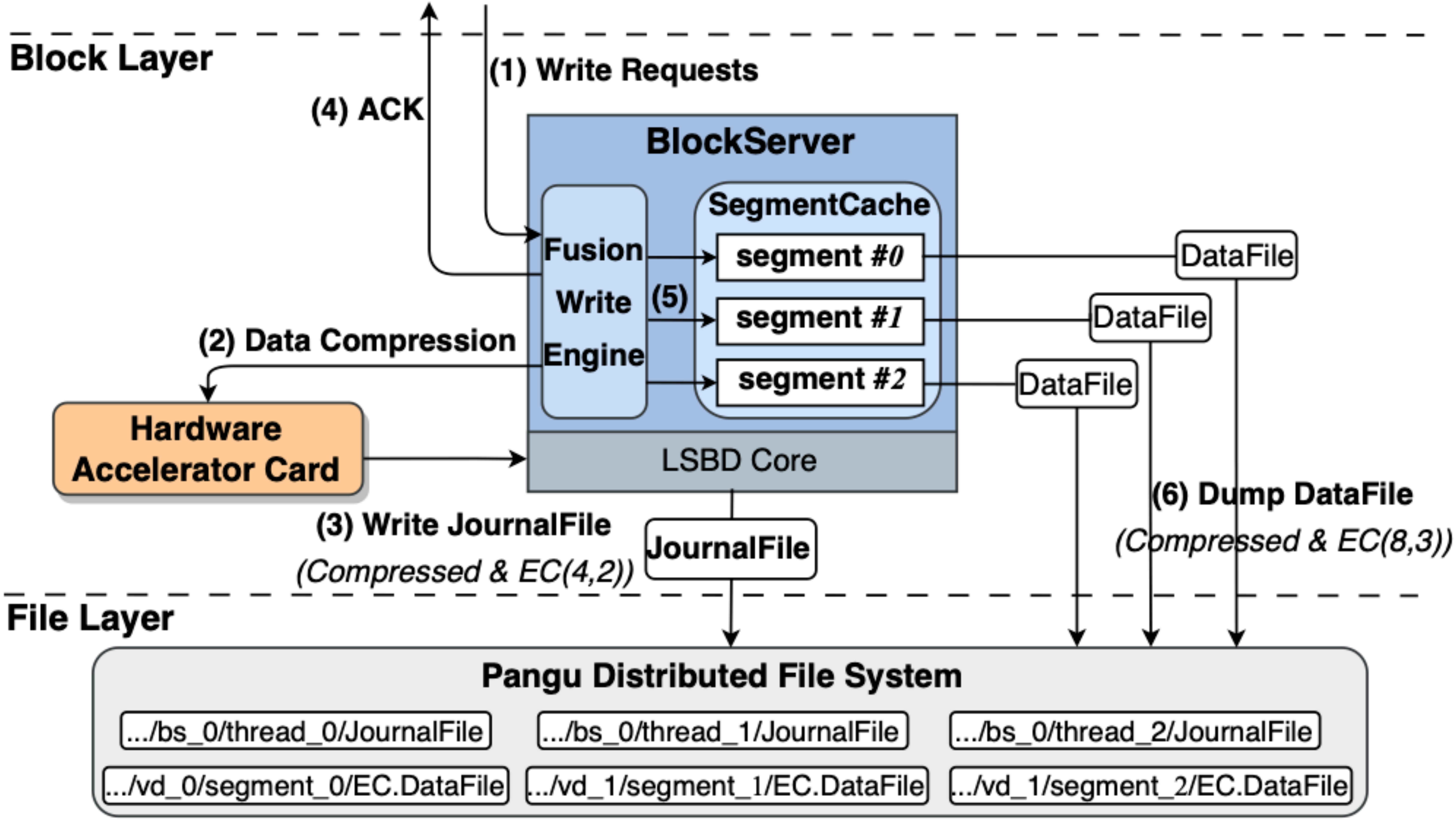
- Max IOPS 1M (10**6) — **50x** compared to EBS1
- Max throughput 4000 MiB/s — **13x** compared to EBS1
- Heavy **network amplification of 4.69x** — compared to 3x in EBS1
- Average space amplification of **1.29x** — compared to 3x in EBS1

EBS3: Reducing Network Amplification

EBS3 Overview

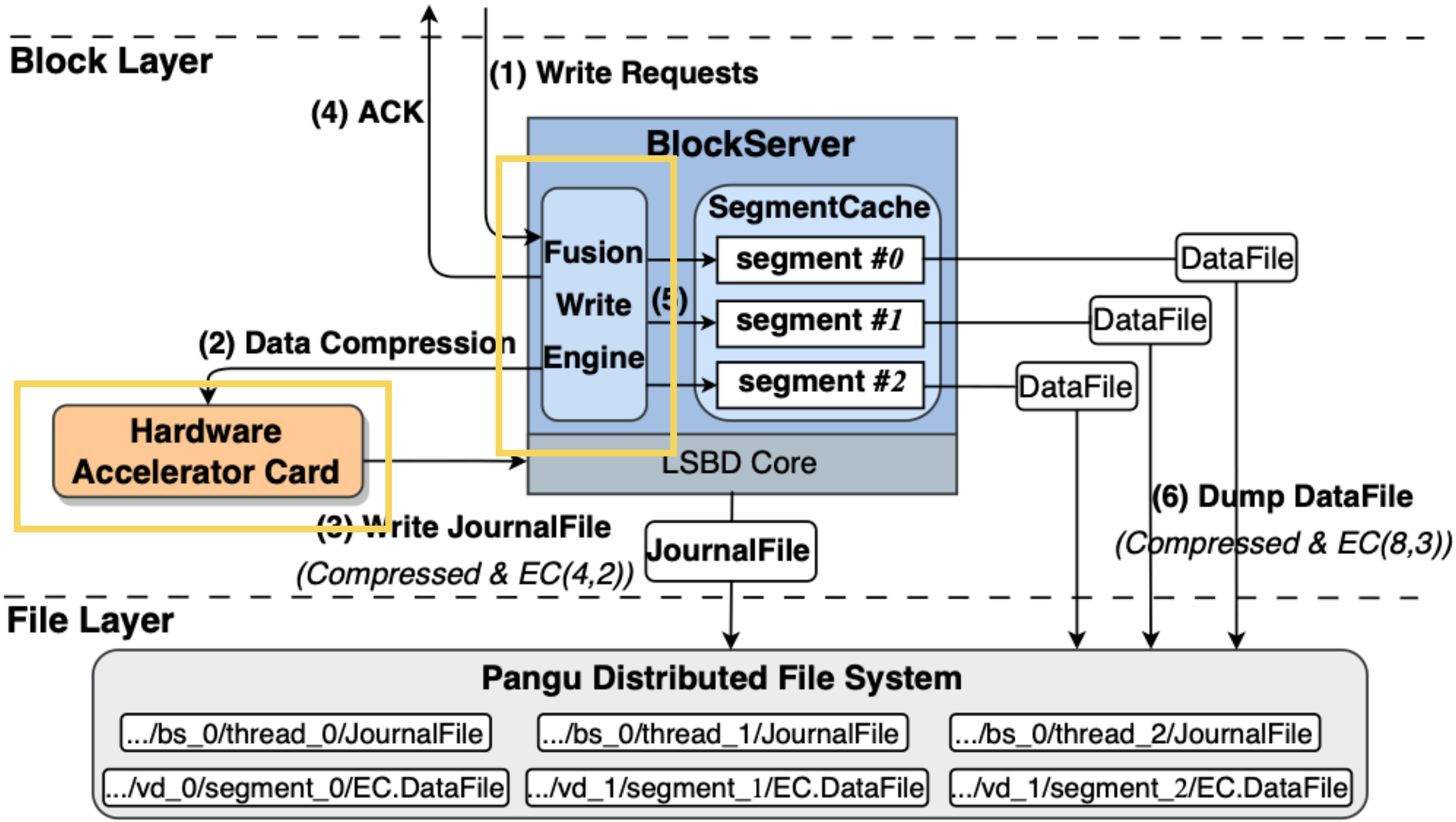
- Adds **compression** and **Erasure Coding** (EC) on the write path
- Batches small writes with **Fusion Write Engine** (**FWE**)
- **Uses FPGA** to offload compression from CPU
- Network amplification **~1.59x** (drops from 4.69x)
- Space amplification **~0.77x**
- **7.3 GiB/s** throughput per card

Write



Block Layer
(1) Write Requests
(4) ACK
BlockServer
SegmentCache
Fusion
Write
Engine
(5)
segment #0
segment #1
segment #2
DataFile
DataFile
DataFile
(2) Data Compression
Hardware Accelerator Card
LSBD Core
(3) Write JournalFile
(Compressed & EC(4,2))
JournalFile
(6) Dump DataFile
(Compressed & EC(8,3))
File Layer
Pangu Distributed File System
.../bs_0/thread_0/JournalFile
.../bs_0/thread_1/JournalFile
.../bs_0/thread_2/JournalFile
.../vd_0/segment_0/EC.DataFile
.../vd_1/segment_1/EC.DataFile
.../vd_1/segment_2/EC.DataFile

Write



Block Layer
(1) Write Requests
(4) ACK
BlockServer
SegmentCache
Fusion
Write
Engine
(5)
segment #0
segment #1
segment #2
DataFile
DataFile
DataFile
(2) Data Compression
Hardware Accelerator Card
LSBD Core
(3) Write JournalFile
(Compressed & EC(4,2))
JournalFile
(6) Dump DataFile
(Compressed & EC(8,3))
File Layer
Pangu Distributed File System
.../bs_0/thread_0/JournalFile
.../bs_0/thread_1/JournalFile
.../bs_0/thread_2/JournalFile
.../vd_0/segment_0/EC.DataFile
.../vd_1/segment_1/EC.DataFile
.../vd_1/segment_2/EC.DataFile

EBS3: Evaluation

- **4,000 MiB/s** throughput and **1M IOPS** per VD which is **13x** and **50x** higher than EBS1
- **Huge performance improvements** over EBS1 in FIO microbenchmark, RocksDB with YCSB and MySQL with Sysbench application workloads

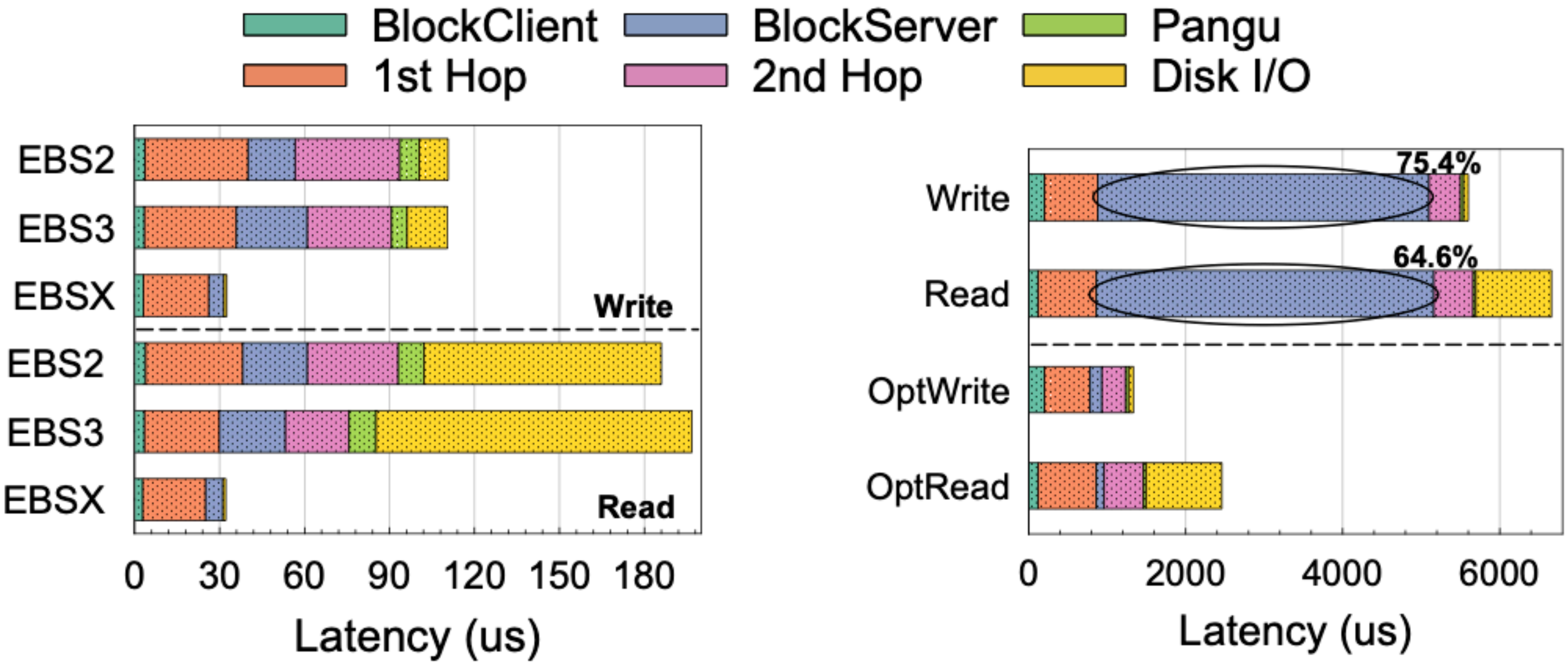
EBS3: Elasticity

Elasticity: 4 Metrics

- Latency both average and 99.999th %ile
- Throughput and IOPS
- Capacity

Elasticity: Latency

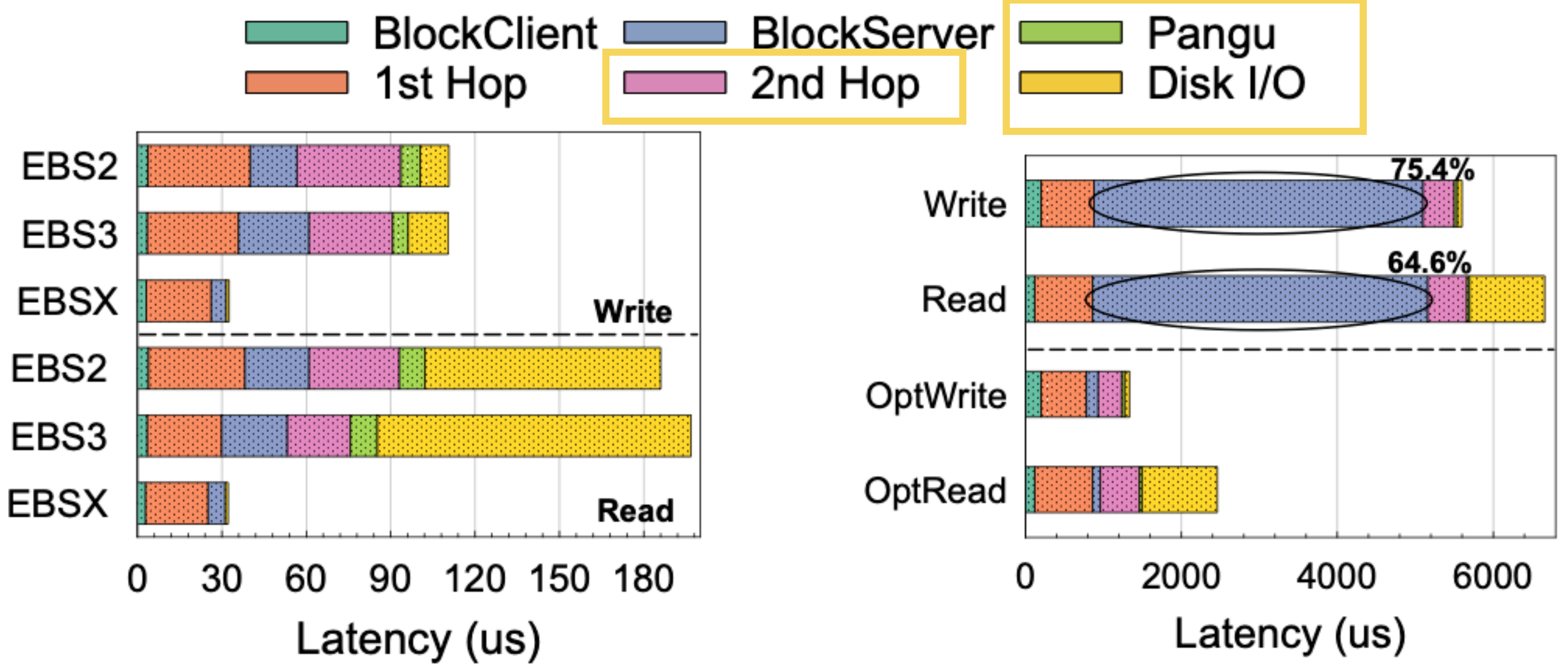
Latency: Average and 99.999th %ile



(a) Average Latency Breakdown of EBS2, EBS3 and EBSX

(b) 99.999th Tail Latency Breakdown of EBS3

Latency: Average



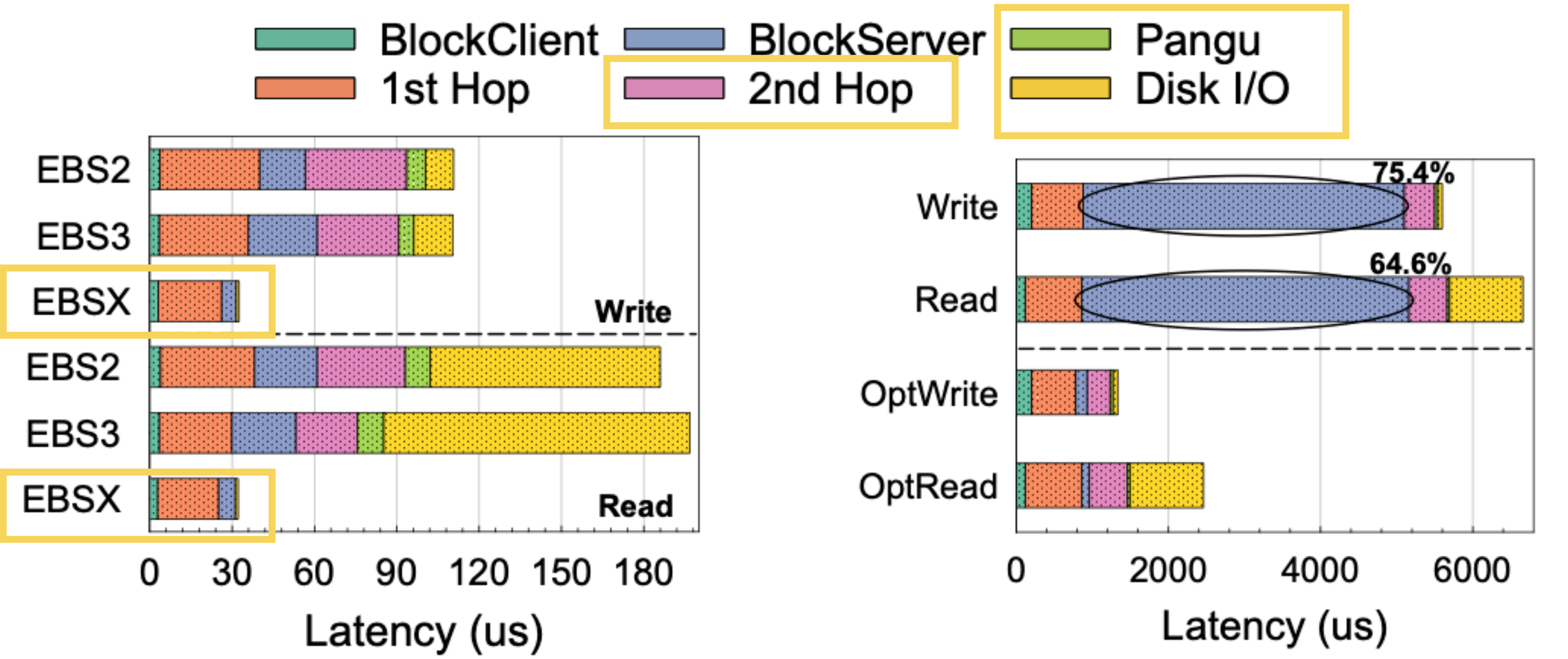
(a) Average Latency Breakdown of EBS2, EBS3 and EBSX

(b) 99.999th Tail Latency Breakdown of EBS3

Average Latency: EBSX

- Mostly in the **hardware** (network + disk)
- Developed **EBSX** — storing data in **PMem** and **skipping 2nd hop** to Pangu
- Data in PMem **eventually flushes** to Pangu

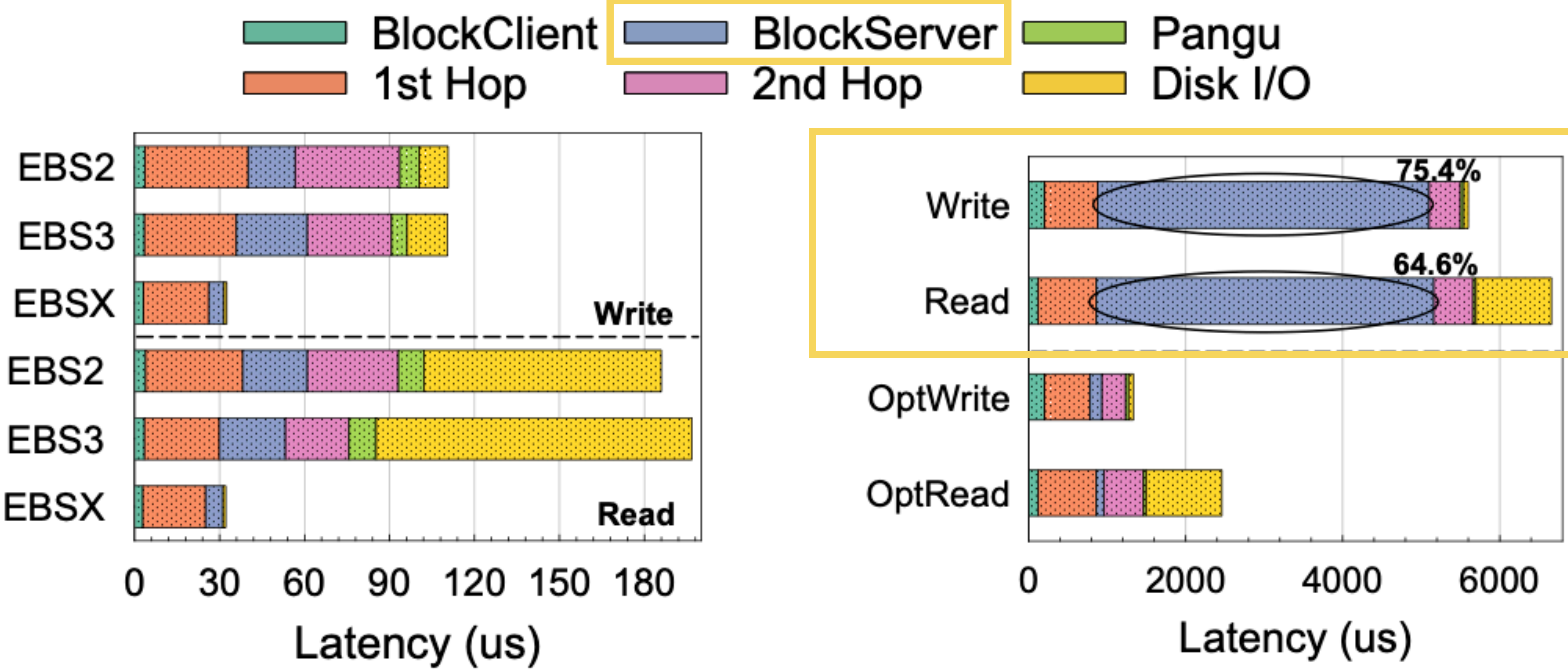
Latency: Average



(a) Average Latency Breakdown of EBS2, EBS3 and EBSX

(b) 99.999th Tail Latency Breakdown of EBS3

Latency: 99.999th %ile



(a) Average Latency Breakdown of EBS2, EBS3 and EBSX

(b) 99.999th Tail Latency Breakdown of EBS3

Tail Latency (99.999th %ile)

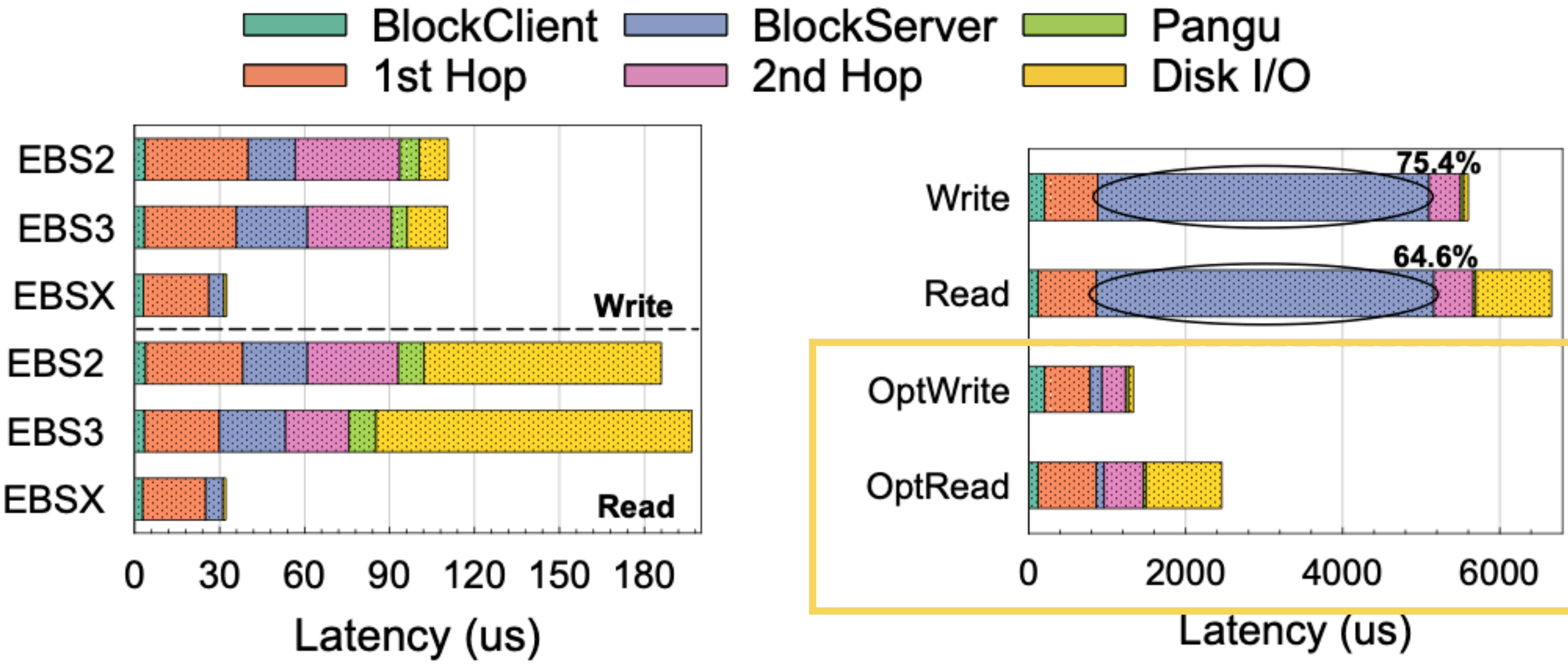
Main causes:

- **Contention** with background tasks (scrubbing, compaction)
- Non-IO RPC destruction in IO thread

Solutions:

- Move background tasks to **a separate thread**
- **Speculative retry** to another replica

Latency: 99.999th %ile



(a) Average Latency Breakdown of EBS2, EBS3 and EBSX

(b) 99.999th Tail Latency Breakdown of EBS3

Elasticity: Throughput and IOPS

Throughput and IOPS: BlockClient

- Move IO processing to the **user space**
- Offload IO to **FPGA**: bypass CPU, CRC calculations, packet transmissions
- **2x100G** network shifts bottleneck to PCIe bandwidth

Throughput and IOPS: BlockServer

- Reduce data sector size to **128KiB** allows **1000 IOPS per 1Gb** (parallelism)
- **Base+Burst** strategy:
  - Priority-based congestion control (Base/Burst priority)
  - Server-wide dynamic resource allocation
  - Cluster-wide hot spot mitigation
- Max **Base** capacity 50K IOPS, max **Burst** 1M IOPS

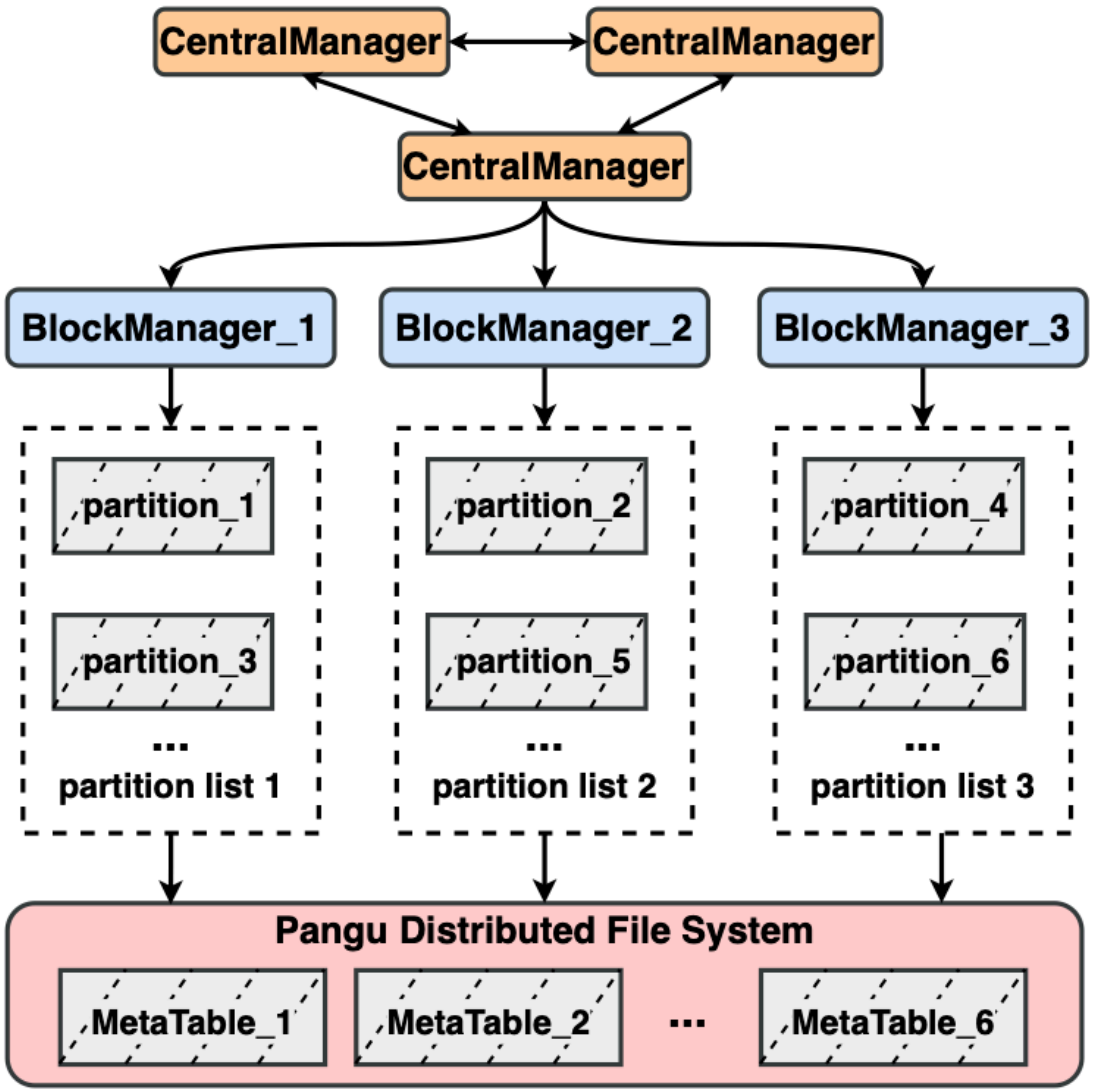
Availability

Availability: Blast Radius

- **Global** — e.g. abnormal behavior of BlockManager
- **Regional** — several VDs, e.g. BlockServer crash. More severe in EBS2 / EBS3 since BlockServer is responsible for more VDs
- **Individual** — single VD. Can cascade into a regional even, e.g. “poison pill”

Availability: Control Plane

Federated BlockManager



Federated BlockManager

- **CentralManager** managers other BlockManagers
- Each BlockManager manages hundreds of VD-level partitions
- On BlockManager failure partitions are **redistributed**

Compare to Millions of Tiny Databases / AWS Physalia.

Availability: Data Plane

Logical Failure Domain

- Address **“poison pill”** problem in software. Core idea is to isolate suspicious segments into a small number of BlockServers
- **Token bucket algorithm** for segment migration. Capacity 3, +1 token every 30 minutes
- Once tokens depleted only migrates to a fixed small (3 nodes) subset of BlockServers — “**Logical Failure Domain**”
- Future failure domains **merge** into one

Conclusions

Conclusions

- **Evolution** of architecture from EBS1 to EBS3 / EBSX
- Discusses **lessons, tradeoffs and various design attempts**
- Talks about **availability, elasticity, hardware offload**

References

References

- **Self reference** for this talk (slides, video, transcript, etc)
  https://asatarin.github.io/talks/2024-05-evolution-of-cloud-block-store/
- Paper "What's the Story in EBS Glory: Evolutions and Lessons in Building Cloud Block Store"
- Millions of Tiny Databases paper

Contacts

- Follow me on Twitter @asatarin
- Follow me on Mastodon https://discuss.systems/@asatarin
- Contact me on LinkedIn https://www.linkedin.com/in/asatarin/
- Watch my public talks https://asatarin.github.io/talks/
- Up-to-date contacts https://asatarin.github.io/about/