2024-10-29

What's the Story in EBS Glory

Evolutions and Lessons in Building Cloud Block Store

Juyong Shin

Networked Systems and Security Lab.

Introduction

- **What's the Story in EBS Glory: Evolutions and Lessons in Building Cloud Block Store**
  - 10 Years of ALIBABA CLOUD EBS Evolution
  - Key Development Lessons
    - Elasticity: Latency, throughput, IOPS, capacity
    - Availability: Failure impact minimization
    - HW Offloading: Tradeoffs & motivations
    - Solution Analysis: Practical considerations

Introduction

- **What’s the Story in EBS Glory: Evolutions and Lessons in Building Cloud Block Store**
  - 10 Years of ALIBABA CLOUD EBS Evolution
  - Key Development Lessons
    - Elasticity: Latency, throughput, IOPS, capacity
    - Availability: Failure impact minimization
    - HW Offloading: Tradeoffs & motivations
    - Solution Analysis: Practical considerations


usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

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

Weidong Zhang, Erci Xu, Qiuping Wang, Xiaolu Zhang, Yuesheng Gu, Zhenwei Lu, Tao Ouyang, Guanqun Dai, Wenwen Peng, Zhe Xu, Shuo Zhang, Dong Wu, Yilei Peng, Tianyun Wang, Haoran Zhang, Jiasheng Wang, Wenyuan Yan, Yuanyuan Dong, Wenhui Yao, Zhongjie Wu, Lingjun Zhu, Chao Shi, Yinhu Wang, Rong Liu, Junping Wu, Jiaji Zhu, and Jiesheng Wu, *Alibaba Group*

https://www.usenix.org/conference/fast24/presentation/zhang-weidong

This paper is included in the Proceedings of the 22nd USENIX Conference on File and Storage Technologies.

February 27–29, 2024 • Santa Clara, CA, USA

978-1-939133-38-0

Open access to the Proceedings of the 22nd USENIX Conference on File and Storage Technologies is sponsored by

NetApp

Introductions

- **Chronological Progession of EBS**
  - EBS Version Evolution
    - EBS1: Computing-storage seperation
    - EBS2: Log structured design & Virtual disk segmentation
    - EBS3: Reduction of traffic amplification

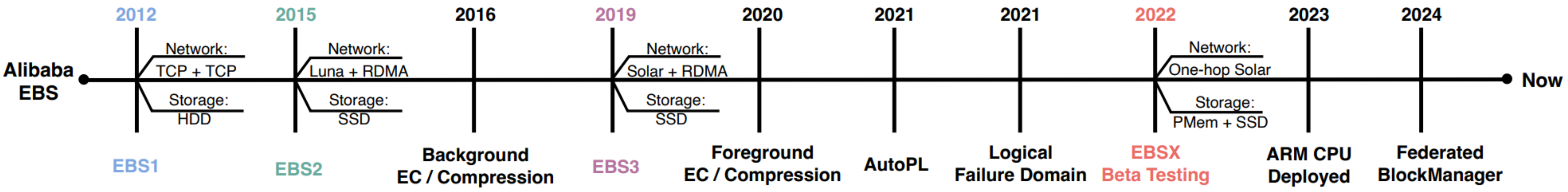


Figure 1: Alibaba EBS Timeline

EBS 1: An Initial Foray

- **Disaggregated Architecture**
  - Separate compute/storage clusters
- **Data Management**
  - Data abstraction with chunks
    - Divide user's VD(Virtual Disk) into 64MiB chunks
    - Replicated into three different ChunkServers
  - Thin provisioning implementation

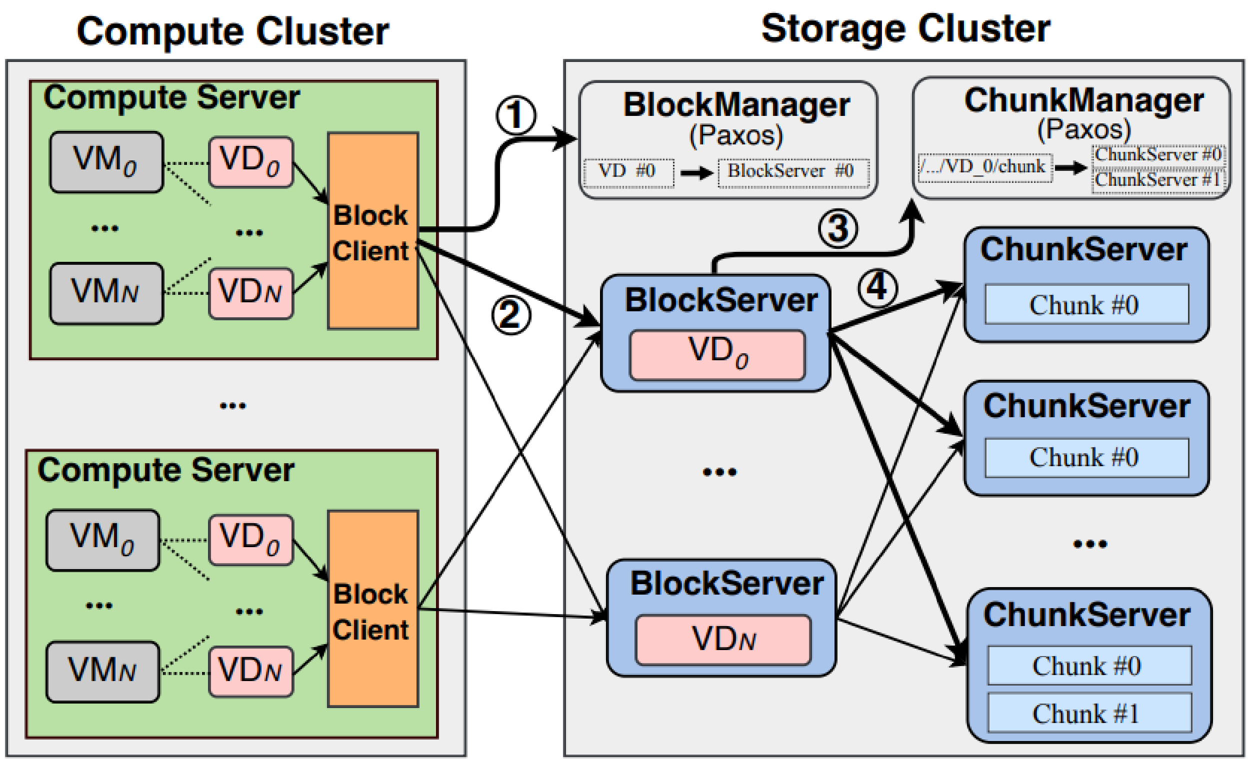


Figure 2: The system architecture of EBS1 (§2.1). ***VD****:Virtual Disk.* ***VM****: Virtual Machine. BlockManagers and ChunkManagers all run three-instance Paxos groups. Each VM can host multiple VDs.*

EBS 1: An Initial Foray

- **Network Architecture**
  - Frontend network
    - Connects compute and storage clusters
  - Backend network
    - Connects BlockServers and ChunkServers
  - Kernel-space networking
- **Technical Limitations**
  - Performance and efficiency challenges
    - Data compression complications with direct mapping
    - Write amplification due to minimum size requirement

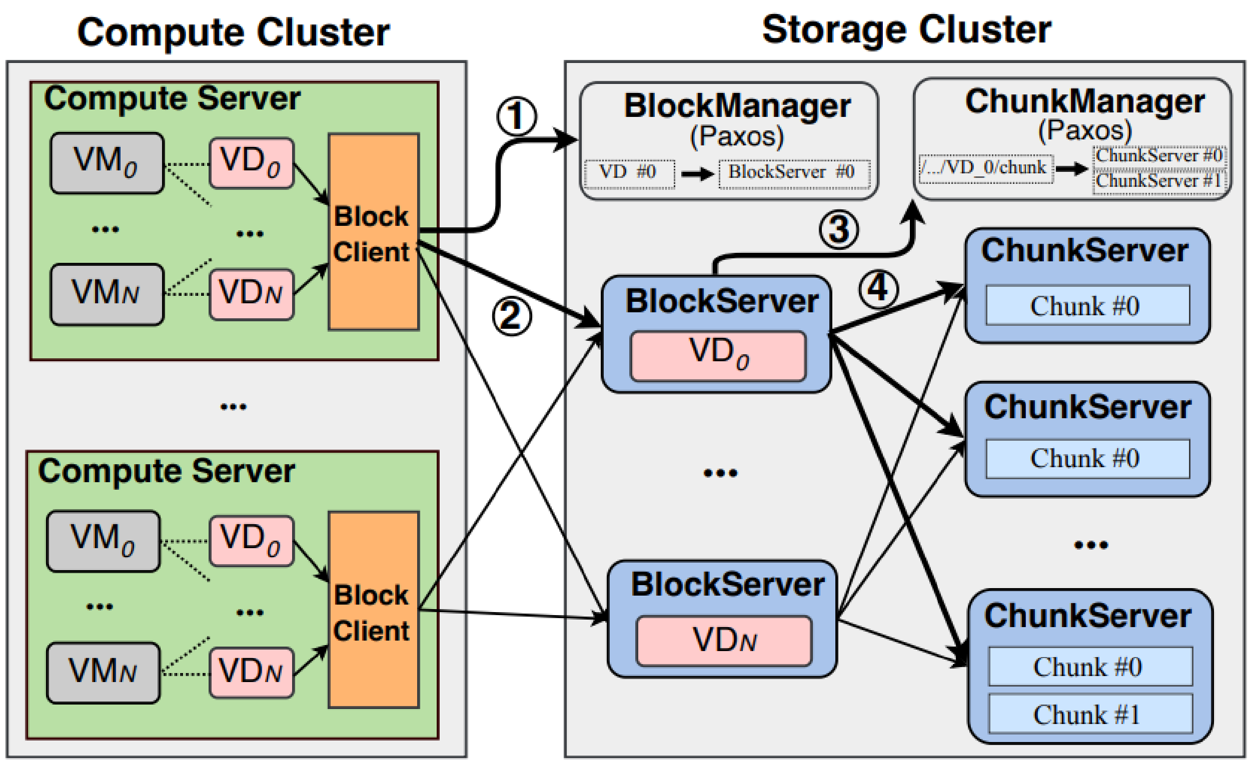


Figure 2: The system architecture of EBS1 (§2.1). ***VD****:Virtual Disk.* ***VM****: Virtual Machine. BlockManagers and ChunkManagers all run three-instance Paxos groups. Each VM can host multiple VDs.*

EBS 2: Speedup with Space Efficiency

- **Storage Management**
  - Delegates data persistence and consensus protocol to Pangu System
    - Append-only file semantics
    - Distributed lock service
- **Log-Structured Design**
  - Translates VD write requests into Pangu append-only writes
    - Enables efficient data compression
    - EC (ErasureCoding) during background garbage collection
- **Segmentation**
  - Multiple BlockServers per VD
  - Segment-level failover
  - BlockManager handles segment migration

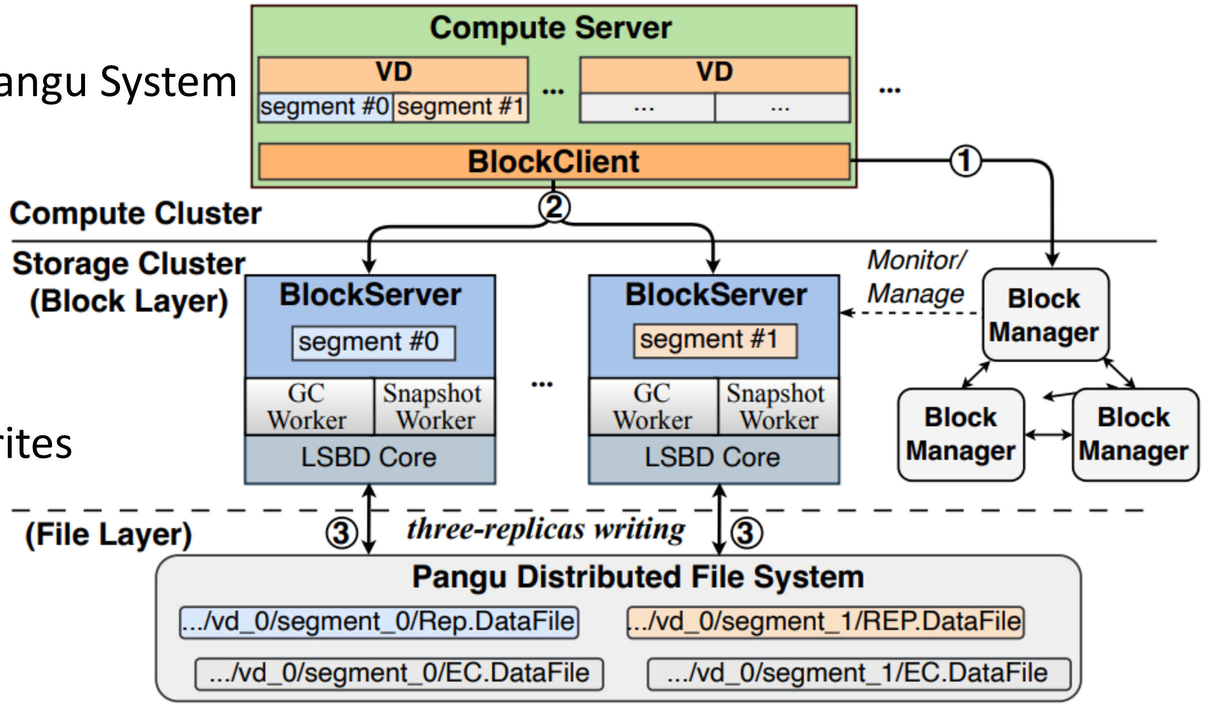


Figure 3: The overview of EBS2. ***LSBD***: *Log-Structured Block Device.* ***REP.DataFile***: *DataFile with three-way replication.* ***EC.DataFile***: *DataFile with EC(8,3) encoding.*

EBS 2: Speedup with Space Efficiency

- **Disk Segmentation**
  - BlockServer operate at the granularity of segments
  - Supports concurrent writes
- **Log-structured Block Device**
  - LSBD Core
    - Supports append-only Semantic
    - Split traffic into front/backend
  - Frontend I/O Flows

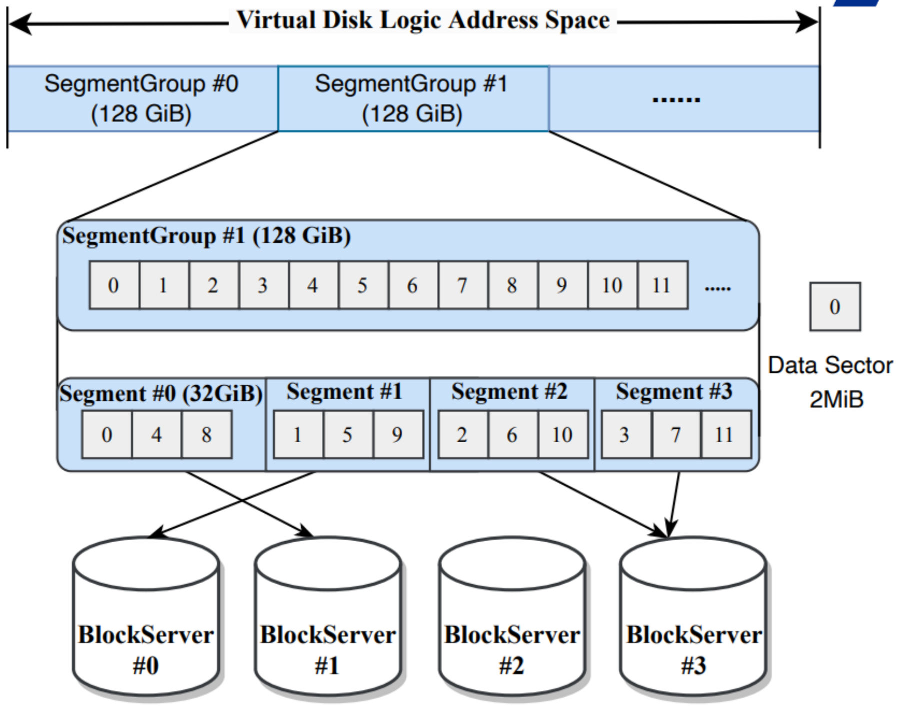


Figure 4: The Disk Segmentation Design of EBS2 (§2.2).

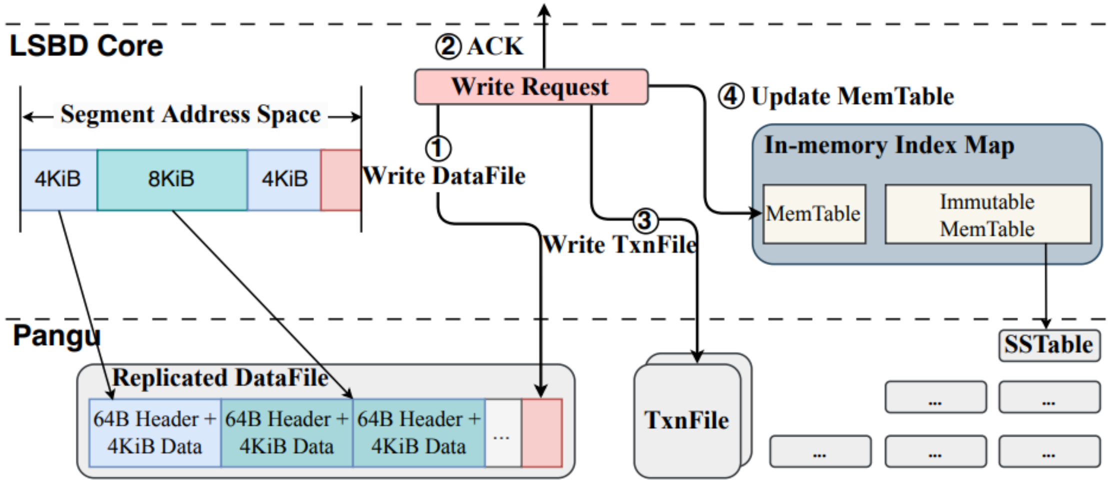


Figure 5: The data organization and persistence format of LSBD. *TxnFile: TransactionFile.*

EBS 2: Speedup with Space Efficiency

- **GC with EC/Compression**
  - Datafile-level Garbage Collection
  - Dynamic GC threshold varies by
    - Cluster storage usage
    - Workload patterns

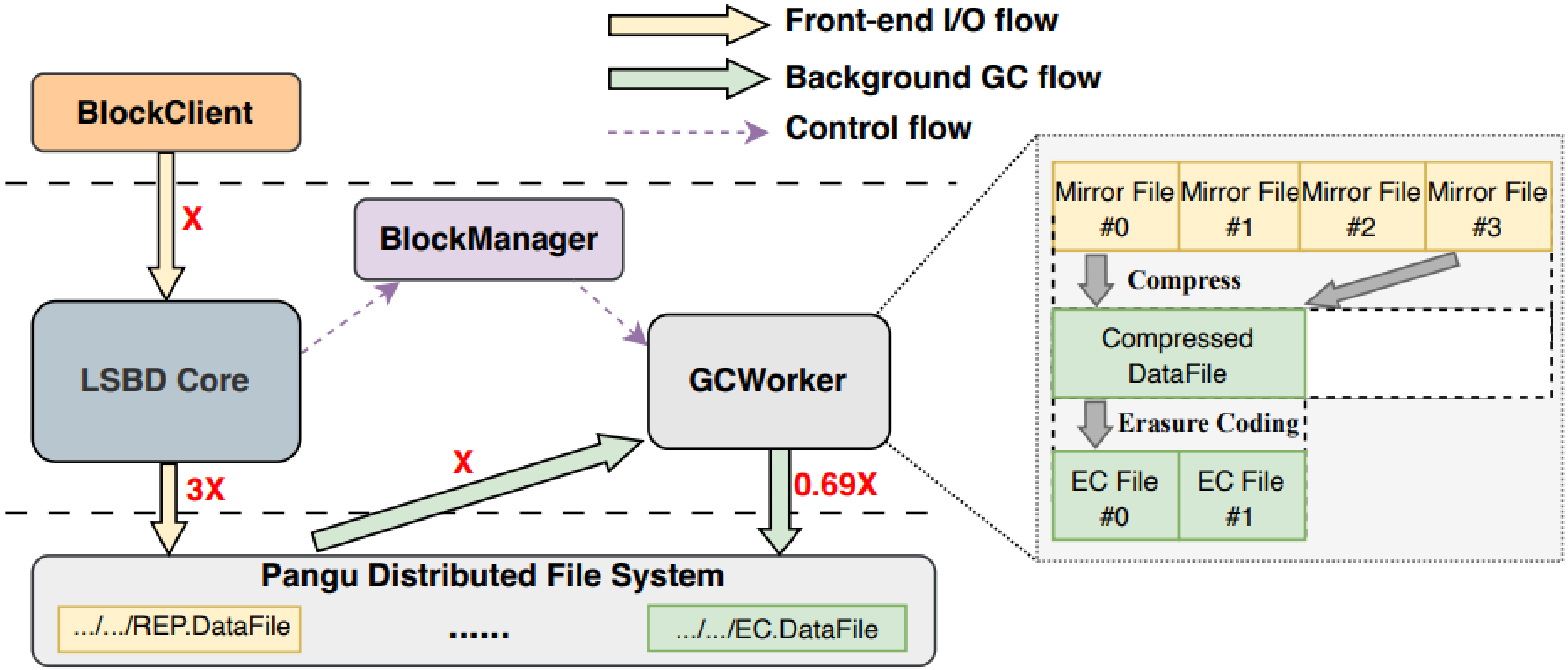


Figure 6: The Garbage Collection in EBS2.

EBS 2: Speedup with Space Efficiency

- **Three Main Components**
  - DataFile Header
    - Start of DataFile
    - Version & checksum
  - CompressedBlocks
    - CompressionHeader
      - Timestamp
      - Compression algorithm
      - Size of CmpBdy
    - CompressionBody(CmpBdy)
      - Compressed data
      - Metadata
  - Offset Table

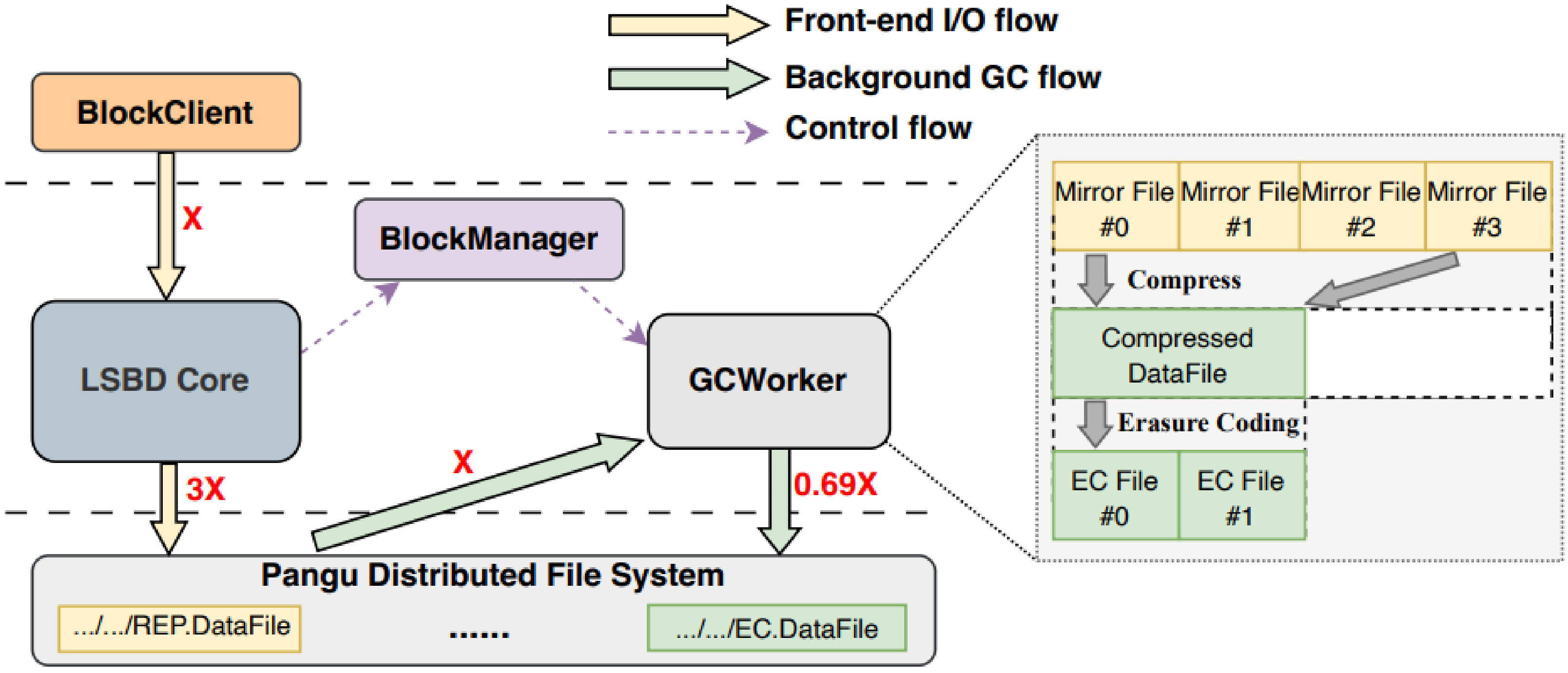


Figure 6: The Garbage Collection in EBS2.

EBS 2: Speedup with Space Efficiency

- **BlockManager with higher availability**
  - Enhanced Availability
    - Survives 2/3 node failures
    - Uses Pangu lock
  - Metadata Management
    - Stored in Pangu files (replicated KV store)
    - vs EBS1: local disk storage with slow repair time

- **Network**
  - Two Fundmental Differences
    - Frontend: User-space TCP implementation(Luna)
    - Backend: 2x25Gbps RDMA network

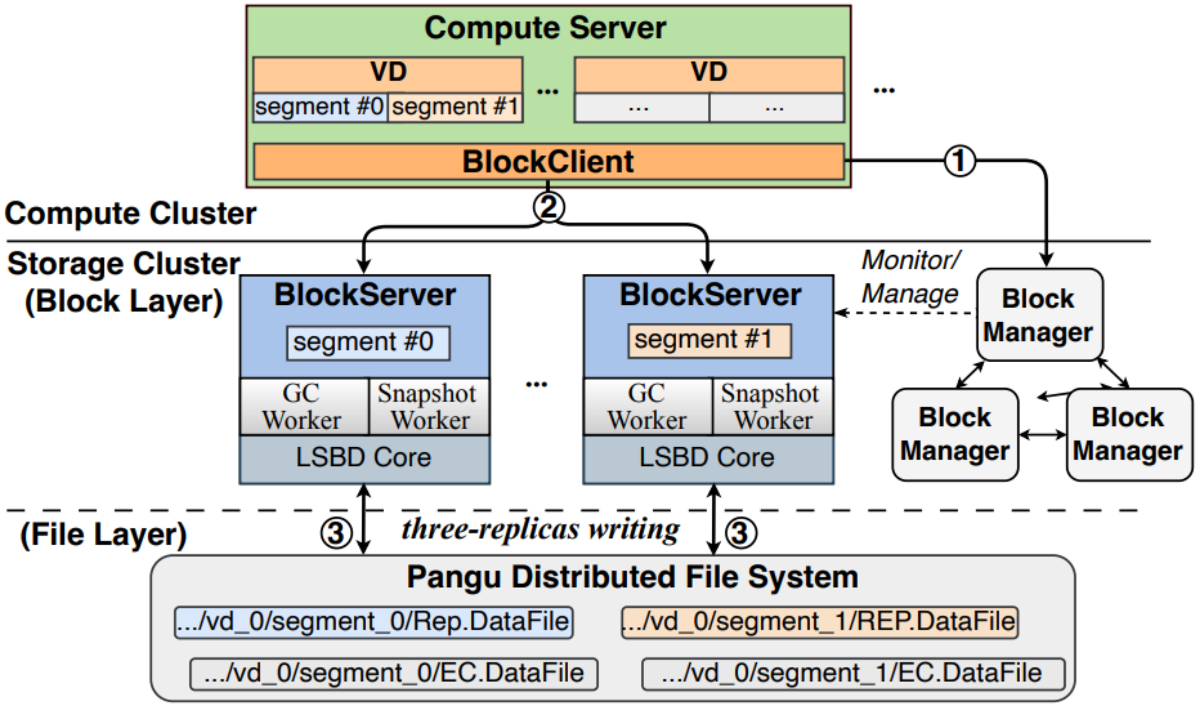


Figure 3: The overview of EBS2. ***LSBD***: *Log-Structured Block Device.* ***REP.DataFile***: *DataFile with three-way replication.* ***EC.DataFile***: *DataFile with EC(8,3) encoding.*

EBS 2: Speedup with Space Efficiency

- **Other Features**
  - Snapshot support
  - Error detection
    - Disk corruption
    - CPU silent data error detection
  - Improved efficiency & performance
- **Limitations**
  - Traffic amplification
    - Three-way replication
    - backend GC
- **Challenges**
  - EC block expansion
  - Data block compression

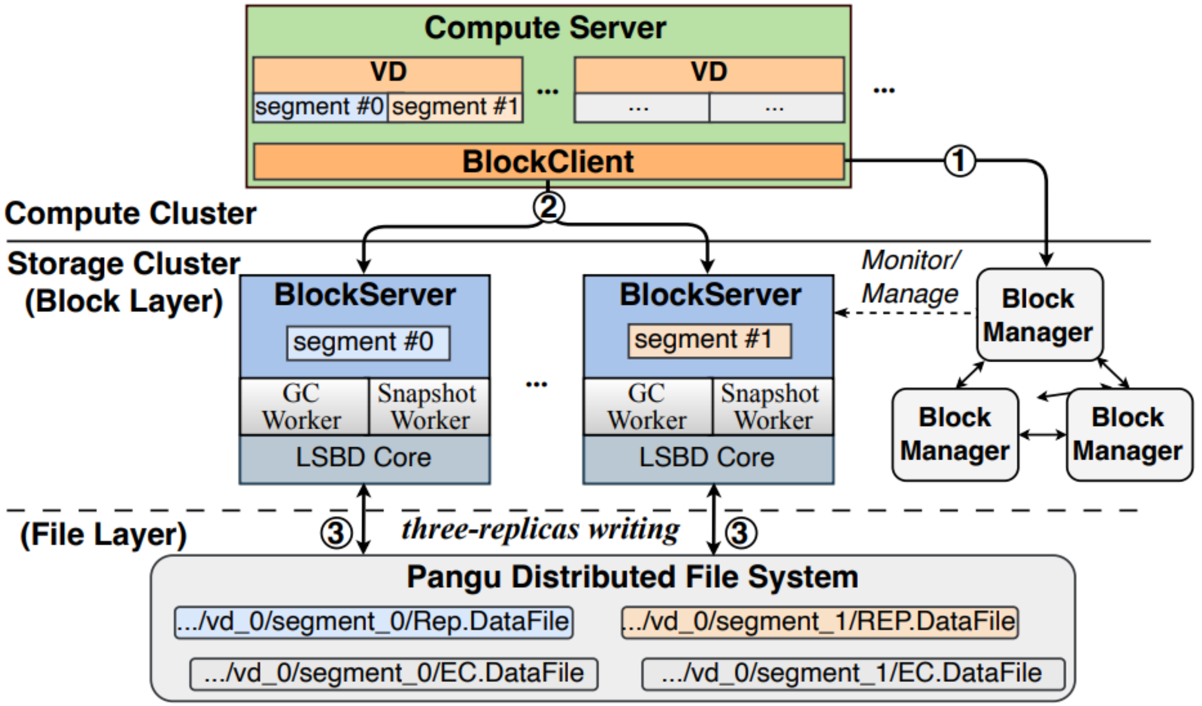


Figure 3: The overview of EBS2. ***LSBD***: *Log-Structured Block Device.* ***REP.DataFile***: *DataFile with three-way replication.* ***EC.DataFile***: *DataFile with EC(8,3) encoding.*

EBS 3: Foreground EC/Compression

- **Fusion Write Engine(FWE)**
  - Merges small writes from different VDs
  - Forms DataBlock

- **FPGA-based Compression**
  - Offloads the compression computations
  - Internal Scheduler
  - Parallel execution units
  - E2E CRC Check
  - Latency and maximum throughput
    - FPGA: 78% latency reduction, 7.3GiB/s max throughput
    - CPU-only: 3.5GiB/s max throughput

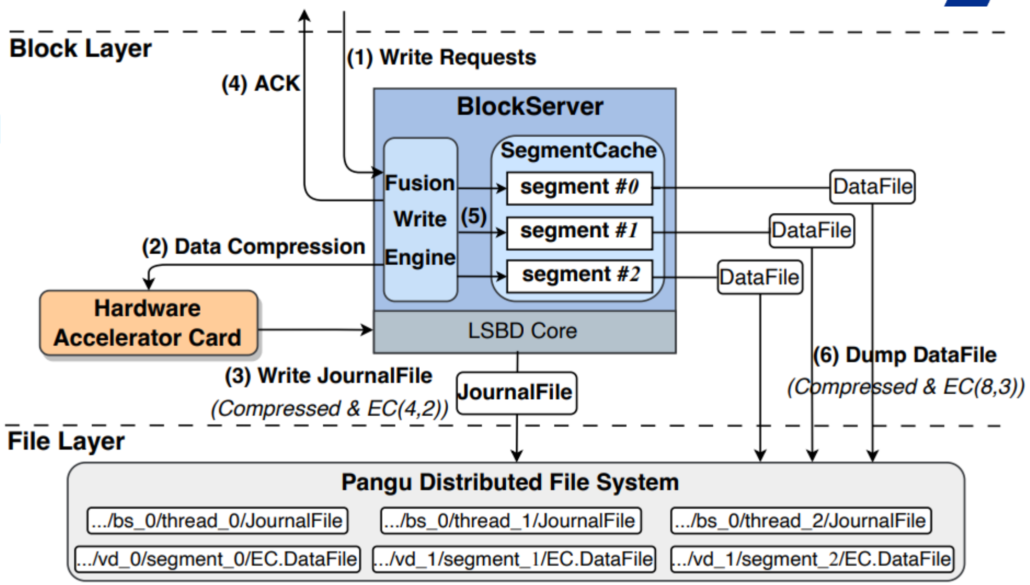


Figure 7: The architecture and I/O flow of EBS3.

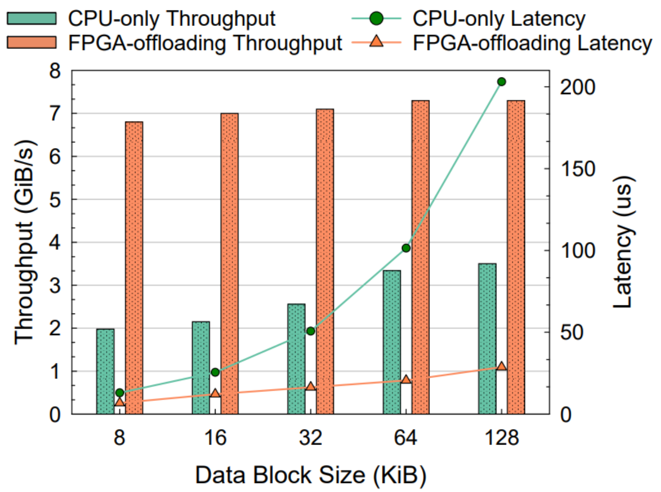


Figure 8: The compression performance comparison of FPGA-offloading and CPU-only with 8 cores compression based on Silesia Compression Corpus.

EBS 3: Foreground EC/Compression

- **Network**
  - Higher linkspeed (2*100 Gbps)
  - Solar: UDP-based transmission protocol
    - HW offloading with DPUs
    - CPU/PCIe bypassing
    - Fast multi-path recovery

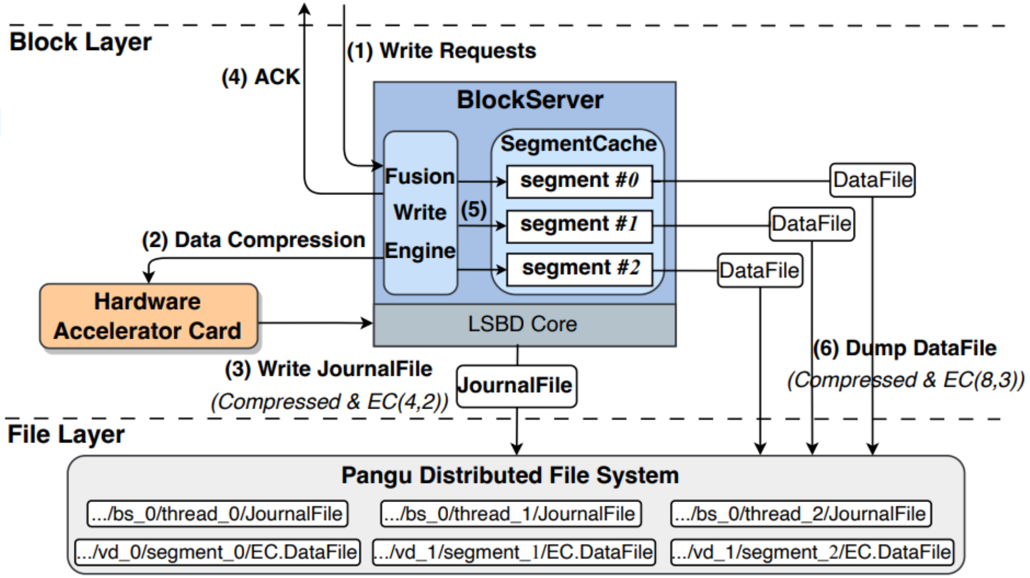
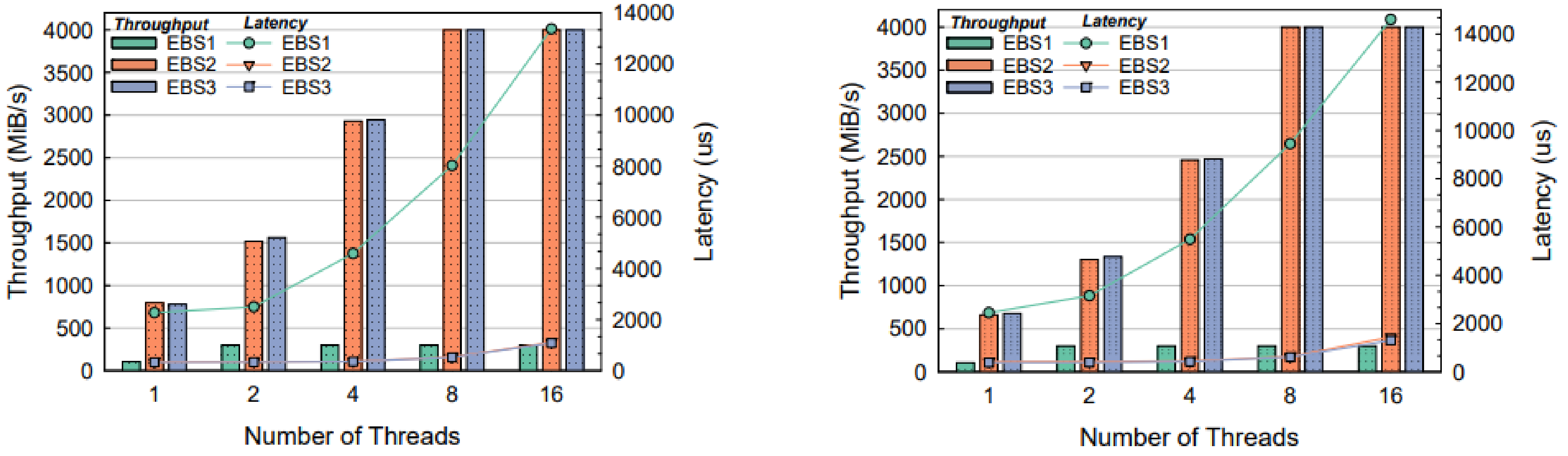


Figure 7: The architecture and I/O flow of EBS3.

- **Deployment & Efficiency**
  - Scale
    - Over 100+ storage clusters
    - 500K+ VDs
  - Efficiency Gains
    - Space Efficiency: 1.29 -> 0.77
    - Traffic amplification: 4.69 -> 1.59



(a) Throughput and Latency of Random **Write** on Thread-to-core Pinning (b) Throughput and Latency of Random **Read** on Thread-to-core Pinning

Figure 9: Random Write/Read Latency of Each Generation EBS under Multiple Threads and 4 KiB-sized I/O. Thread-to-core pinning means that each thread occupies one CPU core exclusively.

Performance Evaluation

- **Microbenchmark (FIO)**
  - Throughput Comparison
    - EBS2/3: 4,000 MiB/s per VD (1M IOPS)
    - 13× higher throughput vs EBS1
    - 50× higher IOPS vs EBS1
  - Scaling Performance
    - Linear increase up to 8 threads
    - Stable latency until 8 threads

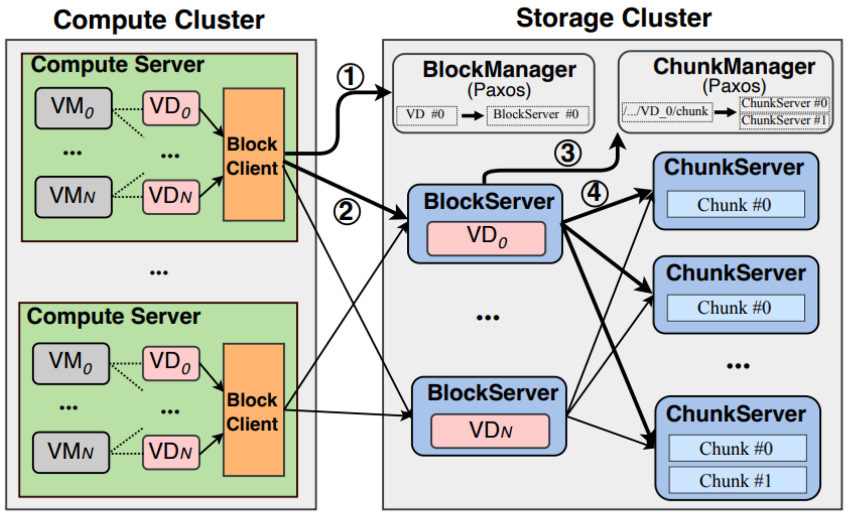


Figure 2: The system architecture of EBS1 (§2.1). ***VD****:Virtual Disk.* ***VM****: Virtual Machine. BlockManagers and ChunkManagers all run three-instance Paxos groups. Each VM can host multiple VDs.*

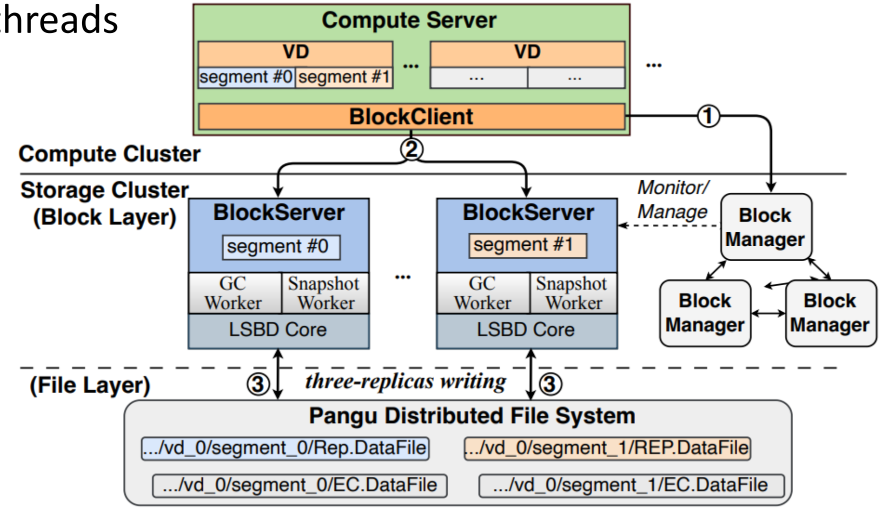


Figure 3: The overview of EBS2. ***LSBD****: Log-Structured Block Device.* ***REP.DataFile****: DataFile with three-way replication.* ***EC.DataFile****: DataFile with EC(8,3) encoding.*

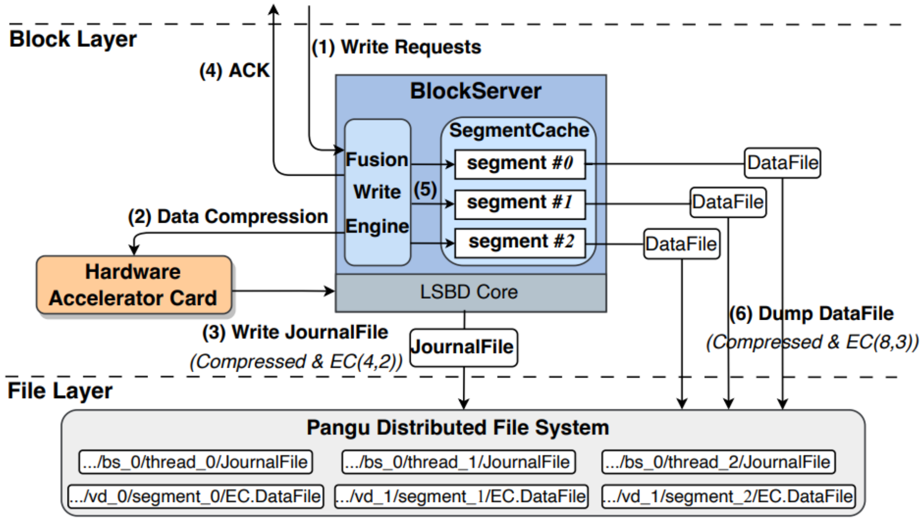
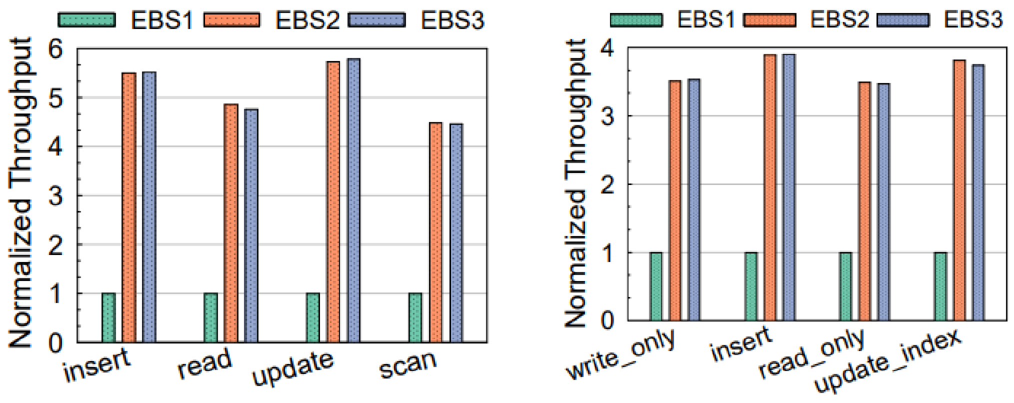


Figure 7: The architecture and I/O flow of EBS3.

Performance Evaluation

- **Application Benchmarks**
  - RocksDB (YCSB)
    - Write workloads: ~550-573% improvement
    - Read workloads: ~470% improvement
  - MySQL (Sysbench)
    - OLTP insert: 389% increase
    - Other workloads: ~350% increase



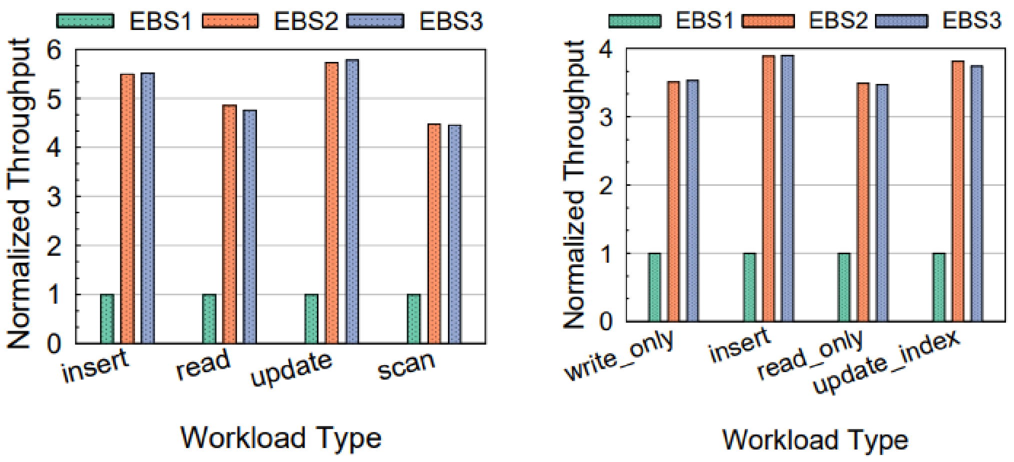
(a) Throughput Comparison of each EBS with **YCSB**

(b) Throughput Comparison of each EBS with **Sysbench**

Figure 10: Throughput Comparison (Normalized with EBS1).

Elasticity: A Tale of Four Metrics

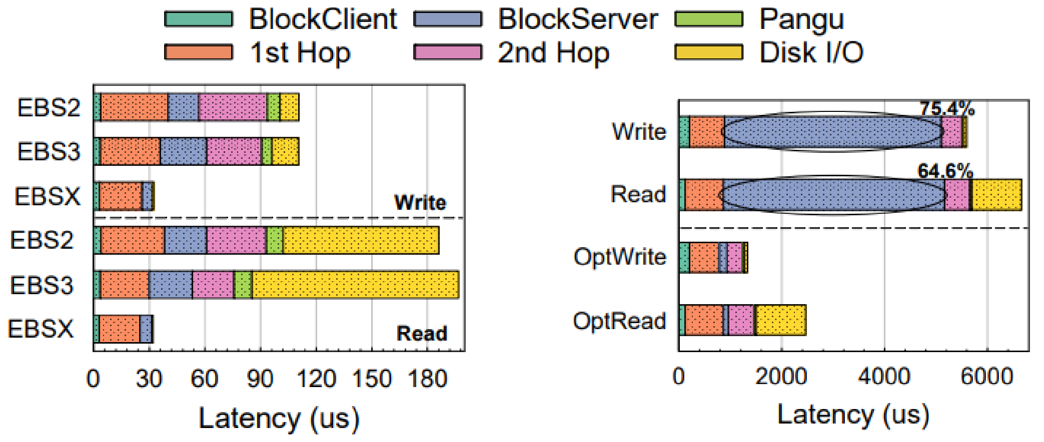
- **Latency**
  - Architecture Impact
    - Determined by architecture & request path
    - Two-hop network structure
    - SW stack processing
    - SSD I/O Time
  - Average Latency Analysis
    - EBS2/3: Depends on HW processing
    - EBSX: 30µs with PMem
  - Tail Latency Improvement
    - Main cause: BlockServer processing
    - Solution: Segregate IO from background tasks
    - Result: Write 1ms, Read 2.5ms (99.999%)



(a) Throughput Comparison of each EBS with **YCSB**

(b) Throughput Comparison of each EBS with **Sysbench**

Figure 10: Throughput Comparison (Normalized with EBS1).



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

(b) 99.999th Tail Latency Breakdown of EBS3

Figure 11: 8 KiB-Sized Avg. and Tail Latency Breakdown of EBS. *1st hop: network latency from compute to storage end. 2nd hop: network latency from BlockServer to Pangu.*

Elasticity: A Tale of Four Metrics

- **Throughput and IOPS**
  - BlockClient
    - Depends on processing and forwarding capability
    - Constrained by network capabilities(2*25 Gbps -> 2*100 Gbps)
  - BlockServer
    - Constrained by the levels of parallelism
    - Reducing the Data Sector size to obtaining higher throughput/IOPS
    - Data Sector: 2MiB ->128KiB
    - Achievement: 1,000 IOPS per GiB

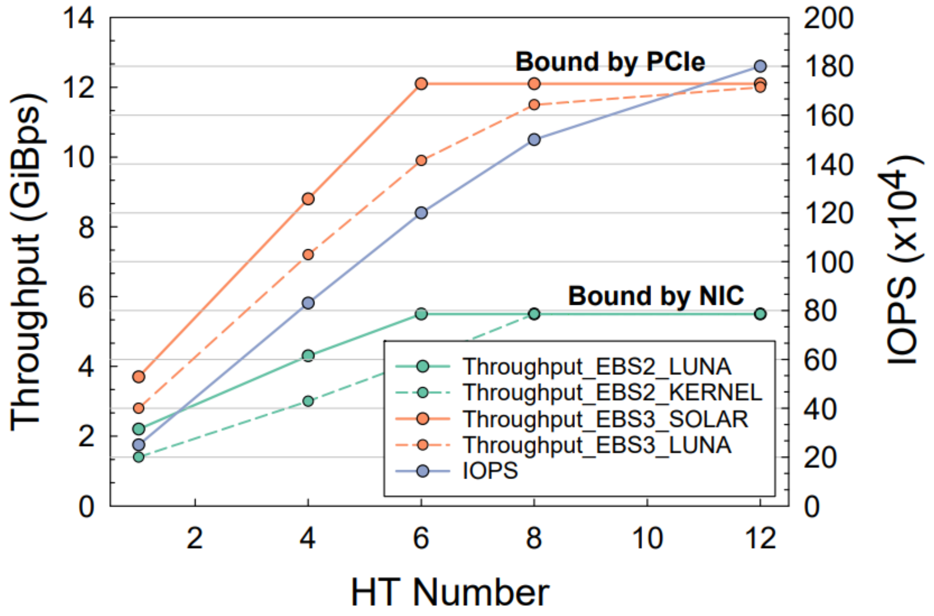


Figure 12: The maximum throughput and IOPS changes of Block-Client with different HT numbers.

Elasticity: A Tale of Four Metrics

- **Throughput and IOPS**
  - Base+Burst allocation
    - Priority-based congestion control
      - BaseIO: Up to 50,000 IOPS (pre-defined)
      - BurstIO: Up to 1M IOPS (based on the available capability)
    - Server-wise dynamic resource allocation
      - Preempt resources from background tasks
    - Cluster-wise hot-spot mitigation
      - Cluster-wise load-balancing

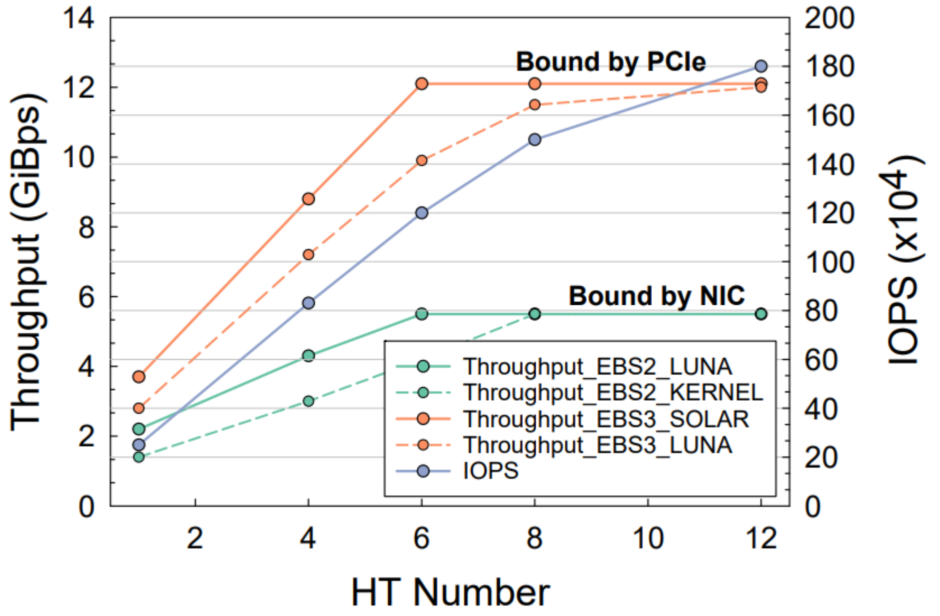


Figure 12: The maximum throughput and IOPS changes of Block-Client with different HT numbers.

Elasticity: A Tale of Four Metrics

- **Capacity**
  - Flexible space resizing
    - Seamless support for VD resizing
    - Range: 1GiB to 64TiB
  - Fast VD cloning
    - Uses the Hard Link of Pangu files
      - Allows the cloning of multiple disks within a storage cluster
    - Performance: 10,000 VDs (40GiB) in 1 Min.

Availability: The Dark Side of Scaling

- **Blast Radius**
  - Global
    - Impacts entire cluster
  - Regional
    - Components to deny service
  - Individual
    - Single VD impacts

  - Mitigation approach
    - Setting smaller cluster
      - EBS reduced the cluster size
      - Benefit: Straightforward and effective
      - Limit: Not effective for regional and individual failure

Availability: The Dark Side of Scaling

- **Control Plane: Federated BlockManager**
  - Initial Challenges
    - Single Leader Issue
    - Single Metadata Table Risk
  - Federated Architecture
    - Multiple BlockManagers per cluster
    - Improved Failure Handling
  - Design Choices
    - Partition-based approach
    - CentralManager Role

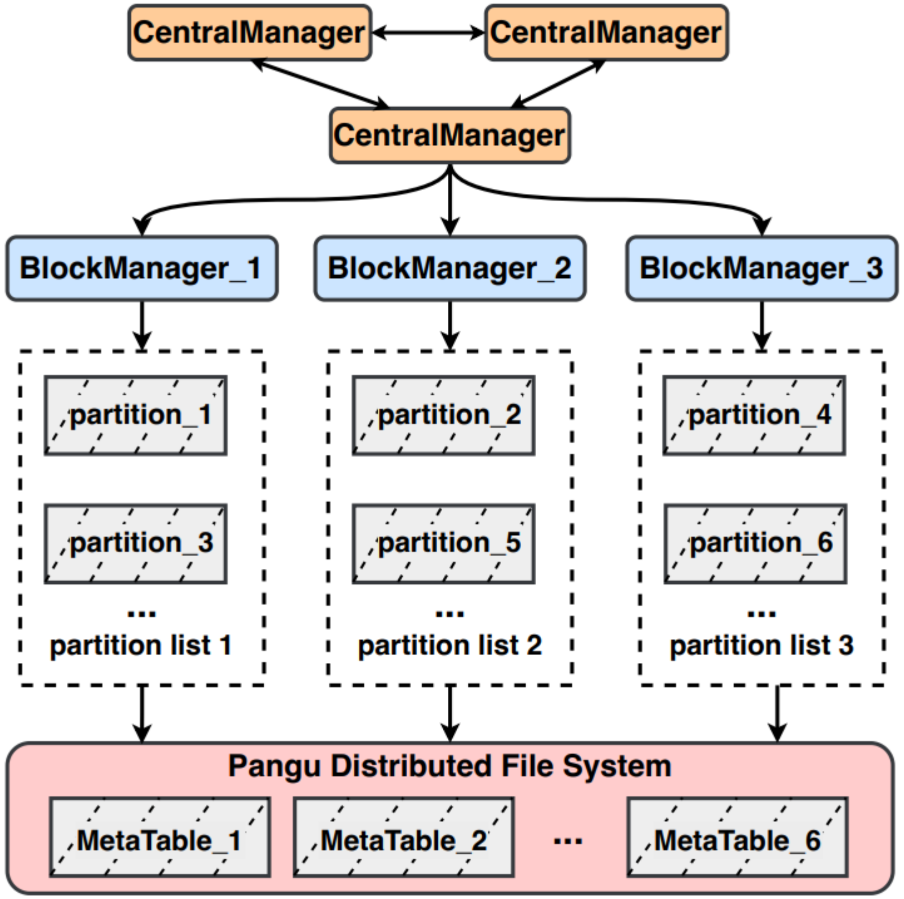


Figure 13: The architecture of Federated BlockManager.

Availability: The Dark Side of Scaling

- **Data Plane: Logical Failure Domain**
  - BlockServer Failures
    - Crash causes segment migration
    - Risk of cascading failures
    - Potential cluster-wide outages
  - Key Observations
    - Failure Characteristics
  - Logical Failure Domain Solution
    - Token bucket system
    - Multi-Failure Handling
    - Results
      - Prevents cluster-wide outages
      - Minimizes impact of error segments
      - Small false negative rate

Availability: The Dark Side of Scaling

- **Lessons Learned**

- Increasing Impact of Node Failures
  - Higher Processing Power and More Users Affected

→

- Solution: Federated Managers
  - Smaller blast radius
  - Maintains cluster scalability

- Forwarding Layer Issues
  - Common in Distributed Services
  - Failure Propagation
    - Request redirection issues
    - Individual -> Regional/Global impact

→

- Solution: Logical failure domain
  - Reactive containment
  - No manual intervention

To Whom the EBS Offloads

- **Offloading BlockClient**
    - BlockClient Bottleneck Issues
        - CPU Limitations
        - ECS(Elastic Computing Service) Requirements
    - Offloading Solutions
        - FPGA Implementation
            - Quick deployment but unstable
        - ASIC Migration
            - Benefits
                - 5% of FPGA CapEx
                - 1/3 power consumption
                - Lower failure rate, …etc
                - Stable functionality

To Whom the EBS Offloads

- **Offloading BlockServer**
  - Initial Compression Challenges
    - LZ4: 25µs latency (25.6% of write)
    - 8 CPU cores for 4,000 MiB/s
  - FPGA Limitations
    - 150 failures per 10K servers
    - Algorithm flexibility issues
  - CPU-based Solution
    - Performance
      - Only 1.3µs higher than FPGA
      - 16 ARM cores match FPGA throughput
    - Benefits
      - No bare-metal restrictions
      - Flexible functionality needs
      - Cost-effective for changes

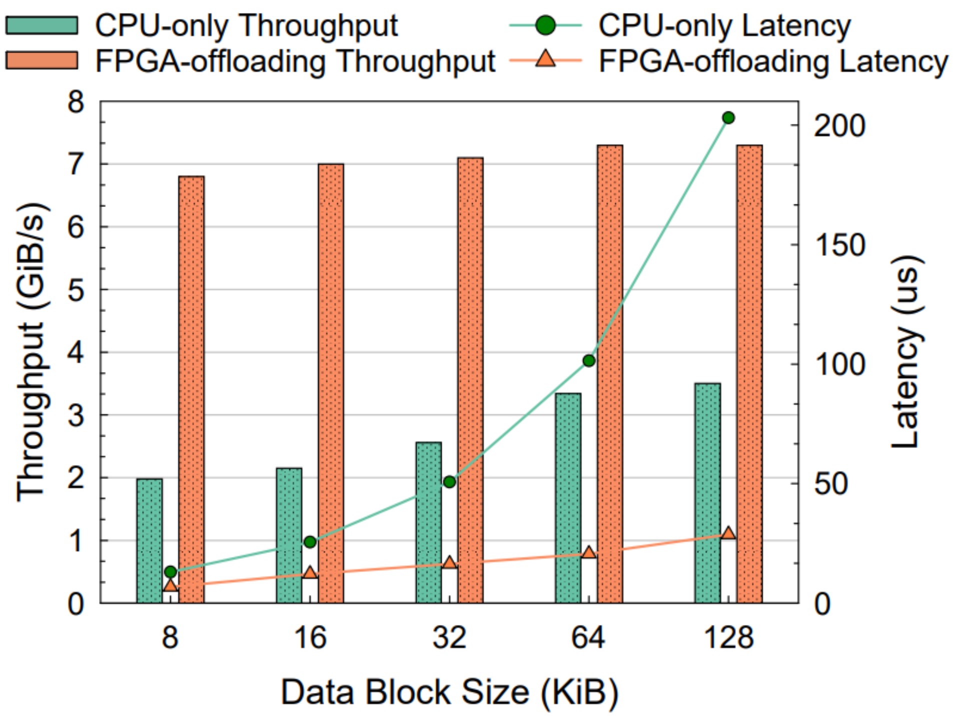


Figure 8: The compression performance comparison of FPGA-offloading and CPU-only with 8 cores compression based on Silesia Compression Corpus.

To Whom the EBS Offloads

- **Field Experience & Lessons**
  - FPGA Experience
    - Initial advantages
      - High flexibility and good performance
    - Limitations
      - Frequent errors, high CapEx, not ideal for large scale
  - ASIC vs ARM Selection
    - ASIC: Compute End
      - Cost-sensitive, Stable operations, Suits massive scale
    - ARM: Storage Backend
      - Frequent upgrades, Low interference, Flexible operations

What If?

- **Q1: What if the log-structured design was never adopted?**
  - Initial Plan for EBS2
    - Attempted: EBS1 + segmentation
    - Result: Abandoned (high engineering cost)
  - Challenge for EBS3
    - Data Aggregation Needs
    - Still have small writes dominance problem
  - Solutions Comparison
    - EBS3: Log-structured approach
      - Merges VD segments and full EC stripes
    - Ceph's Approach
      - Requires cache tier
      - Uses partial writes
      - Higher network overhead

What If?

- **Q2: What if we built EBS with open-source software?**
  - Possible Open-source Approach
    - Possibly use HBase for block layer and HDFS for file layer instead of Pangu
  - EBS Advantages: Co-design
    - Optimized Block Interface
      - Deterministic key space
      - Efficient memory allocation
      - 32GiB segment, 4KiB blocks
  - Performance Optimizations
    - Hardware Offloading
      - FPGA/ARM for compression
    - I/O Path Optimization
      - Separate GC worker, background index compaction
    - Pangu Advantages
      - Userspace file systems, RDMA network, hardware offloading

What If?

- **Q3: What if Pangu and EBS were never separated?**
  - EBS1 Integration Problems
    - Complex Interfaces
      - 10 sets of persistence APIs, hard to maintain
      - 10-month upgrade cycle
  - Benefits of Separation
    - Development
      - Simplified interfaces
      - Faster development
      - Better communication
    - Performance
      - Quick segment operations
      - Isolated failure impact
      - Technology integration ease
      - Pangu's wider usage

Conclusion

- **This paper presents a unique perspective on cloud block storage development**
- **Distinguish itself from both commercial cloud vendors and academic research projects**

- **Key Contributions**
  - How to achieve elasticity across multiple metrics
  - Ways to improve system availability
  - Experiences with hardware offloading strategies
  - Various attempts and alternatives that were considered

Thank you

Juyong Shin (juyongshin@dankook.ac.kr)