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, Jiesheng Wu

Alibaba Group
29 Feb 2024
Background: Elastic Block Store

● EBS
  ✓ VM: Virtual Machine
  ✓ VD: Virtual Disk

● Goal
  ✓ High Performance
  ✓ High Elasticity
  ✓ High Availability

● Compute-Storage Disaggregation
  ✓ VMs and VDs are on different clusters
Evolutions of EBS

Elasticity: A Tale of Four Metrics

Other Topics
EBS1: an Initial Foray

- **Design Goals**
  - **Straightforward** design for fast development/deployment

- **Architecture**
  - VD space is partitioned into fixed-size **Chunks** (64 MiB)
  - Two-layer: Blockserver + Chunkserver
  - Each Chunk is an **Ext4 file**

- **Features**
  - **In-place** updates: VD = Ext4 files
  - N(VDs)-to-1(blockserver) binding
EBS1: An Initial Foray

- **Deployment**
  - Released in 2012, served over 1 million VDs and stored hundreds of PBs of data across hundreds of clusters

- **Limitations**
  - N-to-1 mapping leads to a single hot-point bottlenecks and restricts performance
  - In-place updates hinder the implementation of compression and EC, thereby reducing cost-efficiency

---

[Diagram showing the architecture of EBS1 with VDs, BlockServer, ChunkServer, and the process of Uncompressed LBA #0 being compressed and then Erasure Coded.]
EBS2: Speedup with Space Efficiency

- **Design Goals**
  - ✔ High **performance** and high **space efficiency**

- **Key Designs**
  - ✔ Disk segmentation
  - ✔ Log-structured Block Device (LSBD)
  - ✔ GC with EC/Compression

![Diagram of the EBS2 system architecture](image-url)
EBS2: Speedup with Space Efficiency

**Disk Segmentation**

- The entire VD logic space is divided into multiple contiguous SegmentGroups.
- Each SegmentGroup is organized as a series of Data Sectors.
- Data Sectors are allocated to the Segments in a Round-Robin Fashion.
- BlockServers operate at the granularity of Segments.
EBS2: Speedup with Space Efficiency

- Log-structured Block Device
  - DataFile = (4KB data + 64B Header) x N
  - Txnfile for speeding up failover
  - In-memory Index Map for speeding up read.
EBS2: Speedup with Space Efficiency

- **GC with EC/Compression**
  - LSBD splits traffic into **frontend** (i.e., client I/Os) and **backend** (i.e., GC and compression)
  - GC runs at the granularity of **DataFiles**
  - GC converts the “REP.DataFiles” to “EC.DataFiles” with EC(8, 3) and LZ4/ZSTD compression algorithms

![Diagram of EBS2 architecture]

\[
\text{SpaceCost}_{EBS1} = 3
\]

\[
\text{SpaceCost}_{EBS2} = 1(\text{original}) \times 0.5(\text{compressed}) \times \frac{8+3}{8} \text{(EC)} = 0.69
\]
EBS2: Speedup with Space Efficiency

**Deployment**
- 🔄 100μs avg. write latency and 1 million IOPS per VD.
- ✔ Over 500 clusters and served for 2 million VDs.
- ✔ Low to 1.29 data replicas.

**Limitations**
- ✔ Traffic amplification up to 4.69.
- ✔ As the cost per GiB of SSD decreases, cloud storage has shifted from space-sensitive to traffic-sensitive.

\[
\text{Traffic Amplification}_{EBS1} = 3x \div x = 3
\]

\[
\text{Traffic Amplification}_{EBS2} = (3x + x + 0.69x) \div x = 4.69
\]
EBS2 with Foreground EC/Compression?

- **Fragmented requests prevent Online Compress-EC**
  - EC requires the raw data blocks to typically be at least **16KB**
  - Nearly **70%** of write requests are smaller than 16KB
  - Waiting for merging incurs **extra latency (ranging from 10us to 100ms)**

- **CPU-based compression is slow**
  - 16KB-sized data blocks compression = **25us** for CPUs
  - **CPU resource contention** leads to lower throughput
EBS3: Foreground EC/Compression

**Design Goals**
- Lower traffic consumption and storage space costs
- No performance loss

**Key Designs**
- Bifurcated write path
- Fusion Write Engine
- FPGA-based compression offloading

**Deployment**
- Over 100 clusters for 500K VDs
- Data replicas reduced to 0.77
EBS3: Foreground EC/Compression

● Design Goals
  ✓ Lower traffic consumption and storage space costs
  ✓ No performance loss

● Key Designs
  ✓ Fusion Write Engine
  ✓ FPGA-based compression offloading
  ✓ Traffic reduced from 4.69 to 1.59

● Deployment
  ✓ Over 100 clusters for 500K VDs
  ✓ Data replicas reduced to 0.77

\[
\text{Traffic Amplification}_{EBS2} = (3x + x + 0.69x) \div x = 4.69
\]
\[
\text{Traffic Amplification}_{EBS3} = (0.9x + 0.69x) \div x = 1.59
\]
## Comparison of Three Generations of EBS

<table>
<thead>
<tr>
<th></th>
<th>EBS1</th>
<th>EBS2</th>
<th>EBS3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Avg. Latency</strong></td>
<td>Millisecond Level</td>
<td>Hundred-microsecond Level</td>
<td>Hundred-microsecond Level</td>
</tr>
<tr>
<td><strong>MAX. IOPS / Throughput</strong></td>
<td>25,000</td>
<td>1,000,000</td>
<td>1,000,000</td>
</tr>
<tr>
<td><strong>Key Features</strong></td>
<td>In-place updates N-to-1mapping</td>
<td>Background EC &amp; Compression</td>
<td>Foreground EC &amp; Compression</td>
</tr>
<tr>
<td><strong>Space Cost (Replicas per Data)</strong></td>
<td>3</td>
<td>1.29</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>Traffic Amplification</strong></td>
<td>3</td>
<td>4.69</td>
<td>1.59</td>
</tr>
</tbody>
</table>
Evolving Journey of EBS

Elasticity: A Tale of Four Metrics

Other Topics
Metrics #1: Latency

- Elasticity of latency is coarse-grained
  - Defined by the architectures

- EBSX
  - Shorten the path (e.g., skip a network hop)
  - Use faster devices (e.g., PMem instead of SSD)
  - Simple and efficient data consistency protocol

- Tail latency
  - Software-induced tail latency can be the dominant
  - Separate client IOs from background tasks (e.g., GC)

![Latency Graph]
Metrics #2 & #3: IOPS and Throughput

- Upper bound is determined by BlockClient
  - Backend can be easily extended
  - BlockClient is bound by processing and forwarding capability
  - From kernel-space to user-space, then to hardware offloading

- High IOPS/Throughput is often desired but not always needed
  - Auto performance level (AutoPL) Virtual Disk: on demand without altering the capacity
  - Base + Burst strategy: efficiently allocating IOPS/throughput to VDs
  - Base throughput means can definitely be satisfied
  - Burst throughput means trying my best to satisfy
Metrics #4: Capacity

- **Flexible space resizing**
  - Achieve resizing via adding or removing *SegmentGroups*
  - Virtual disk sizes up to 64 TiB

<table>
<thead>
<tr>
<th>Virtual Disk Logic Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegmentGroup #0 (128 GiB)</td>
</tr>
<tr>
<td>SegmentGroup #1 (128 GiB)</td>
</tr>
<tr>
<td>......</td>
</tr>
</tbody>
</table>

- **Fast VD cloning**
  - *Hard Link* of Pangu files
  - Up to 10,000 virtual disks (each 40 GiB) in 1 min
Evolving Journey of EBS

Elasticity: A Tale of Four Metrics

Other Topics
Other Topics

Availability Threats and Solutions  (See Section 4)

- **Challenge 1:** a BlockServer crash impacts more VDs
  - Solution: Federated BlockManager (Two-layer control nodes)
- **Challenge 2:** Segment migration leads to cascading failures
  - Solution: Logical Failure Domain (Limited migration)

EBS Offloading  (See Section 5)

- FPGA is **not ideal:** expensive, high failure rates
- Blockclient offloading: FPGA → ASIC: 1. cost-friendly 2. a fixed set of functions
- BlockServer offloading: FPGA → Many-core ARM: 1. cost-friendly 2. comparable performance

What if?  (See Section 6)

- Q1: W/o log-structured design? Both cost and performance cannot move forward.
- Q2: EBS with open-source software? Co-design will be never possible.
- Q3: Not separating Pangu? Slow down the development of EBS.

Please see more details in the paper.
Thanks

Q & A

Contact: zhangweidong.zwd@alibaba-inc.com / iszhangwd@hotmail.com