MOS: Taming the Cloud Object Storage

Ali Anwar*, Yue Cheng*, Aayush Gupta†, Ali R. Butt*

*Virginia Tech  &  †IBM Research – Almaden
Cloud object stores enable cost-efficient data storage
Cloud object store supports various workloads

- Website
- Online video sharing
- Online gaming
- Enterprise backup

Object storage
One size does not fit all

Replace monolithic object store with specialized fine-grained object stores each launched on a sub-cluster
Reason 1: Classification of workloads

Applications have different service level requirements, e.g., average latency per request, queries per second (QPS), and data transfer throughput (MB/s)

- Online gaming
- Online video sharing
- Enterprise backup
Small objects

Website
Get: 90%, Put: 5%, Delete: 5%

Object storage
~ 1-100 KB

Online gaming
Get: 5%, Put: 90%, Delete: 5%
Large objects

Online video sharing
Get: 90%, Put: 5%, Delete 5%

Object storage
~ 1-100 MB

Enterprise backup
Get: 5%, Put: 90%, Delete 5%
Reason 2: Heterogeneous resources

- Dcenters hosting object stores are becoming increasingly heterogeneous
- Hardware to application workload mismatch
- Meeting SLA requirement is challenging
Outline

Introduction
Motivation
Contribution
Design
Evaluation
Background: Swift object store

Object storage = Proxy server = Storage nodes

Account/Container/Object
Swift: Proxy and Storage servers

Object storage = Proxy server

1

Storage nodes

Account/Container/Object

2
Swift: Ring architecture

Object storage = Proxy server

Storage nodes

Account/Container/Object
Benchmark used: CosBench

- COSBench is Intel developed Benchmark to measure Cloud Object Storage Service performance
  - For S3, OpenStack Swift like object store
  - Not for file system or block device system
- Used to compare different hardware software stacks
- Identify bottlenecks and make optimizations
### Workload used

<table>
<thead>
<tr>
<th>Workload</th>
<th>Object Size</th>
<th>Distribution</th>
<th>Application scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workload A</td>
<td>1 – 128 KB</td>
<td>G: 90%, P: 5%, D:5%</td>
<td>Web hosting</td>
</tr>
<tr>
<td>Workload B</td>
<td>1 – 128 KB</td>
<td>G: 5%, P: 90%, D:5%</td>
<td>Online game hosting</td>
</tr>
<tr>
<td>Workload C</td>
<td>1 – 128 MB</td>
<td>G: 90%, P: 5%, D:5%</td>
<td>Online video sharing</td>
</tr>
<tr>
<td>Workload D</td>
<td>1 – 128 MB</td>
<td>G: 5%, P: 90%, D:5%</td>
<td>Enterprise backup</td>
</tr>
</tbody>
</table>
Experimental setup for motivational study

COSBench 32 cores

32 cores 8 cores

1 Gbps 10 Gbps

Proxy servers

Storage servers

3 SATA SSD/node

32 cores
Configuration 1 – Default monolithic

COSBench → Round robin → 32 cores → 1 Gbps

8 cores → 10 Gbps

3 SATA SSD/node → 32 cores
Configuration 2 – Favors small objects

COSBench

8 cores

32 cores

10 Gbps

1 Gbps

Small objects

Large objects

3 SATA SSD/node

32 cores
Configuration 3 – Favors large objects

Small objects

COSBench

1 Gbps

32 cores

Large objects

COSBench

10 Gbps

32 cores

8 cores

3 SATA SSD/node
Performance under multi tenant environment – Workload A & B

- Throughput (QPS)
  - Small objects
    - Config 1
    - Config 2
    - Config 3
  - Large objects
    - Config 1
    - Config 2
    - Config 3

Workload
- A
- B
- C
- D
Performance under multi tenant environment - Workload A & B

Throughput (MB/s)

Small objects

Config 1
Config 2
Config 3

Large objects

A  B  C  D

Workload
Performance under multi tenant environment - latency

Latency (sec)

Small objects
- Config 1
- Config 2
- Config 3

Large objects

Workload
- A
- B
- C
- D

Small objects

Large objects

Latency (sec)
Key Insights

- Cloud object store workloads can be classified based on the size of the objects in their workloads.
- When multiple tenants run workloads with drastically different behaviors, they compete for the object store resources with each other.
Outline

Introduction

Motivation

Contribution

Design

Evaluation
Contributions

- Perform a performance and resource efficiency analysis on major hardware and software configuration opportunities

- We design MOS, Micro Object Storage:
  - 1) dynamically provisions fine-grained microstores
  - 2) exposes the interfaces of microstores to the tenants

- Evaluate MOS to showcase its advantages
Outline

Introduction
Motivation
Contribution
Design
Evaluation
Design criteria for MOS

- We studied the effect of three knobs on performance of a typical object store to come up with design rules/ rules of thumb
  - Proxy Server settings
  - Storage Server settings
  - Hardware changes
Effect of Proxy server settings

Small objects

Large objects
Effect of Proxy server settings

Throughput ($10^3$ QPS) vs. Proxy workers

- QPS
- CPU util

Per-node CPU util (%)

0% 20% 40% 60% 80% 100%

100% util

Small objects

Large objects

10 Gbps NIC bandwidth limit

Throughput (GB/s) vs. Proxy workers

0 0.5 1 1.5 2

0 2 4 8 16 32 2x

Small objects

Large objects
Effect of Storage server settings

![Graph showing the effect of storage server settings on throughput. The x-axis represents the number of object storage workers, ranging from 1 to 32. The y-axis represents throughput in units of (10^3) QPS and GB/s. The graph demonstrates an increase in throughput as the number of object storage workers increases.](image)
Effect of Storage server settings

Throughput (10^3 QPS)

Throughput (GB/s)

Object storage workers

Small objects
Effect of Storage server settings

Large objects

Throughput (10^3 QPS)

Throughput (GB/s)

Object storage workers

QPS

GB/s
Effect of hardware settings

Throughput (10^3 QPS)

Small objects

Throughput (GB/s)

Large objects

Throughput

HDD
SSD

HDD
SSD

1 Gbps 10 Gbps

1 Gbps 10 Gbps
Rules of thumb

- CPU on proxy serves as the first-priority resource for small-object intensive workloads
- Network bandwidth is more important than CPU on proxy for large-object intensive workloads
- $\text{proxyCores} = \text{storageNodes} \times \text{coresPerStorageNode}$
- $\text{BW}_{\text{proxies}} = \text{storageNodes} \times \text{BW}_{\text{storageNode}}$
- Faster network cannot effectively improve QPS for small-object intensive workloads – use weak network (1 Gbps NICs) with good storage devices (SSD)
MOS Design

- Load balancer/Load redirector
- Object storage
- Proxy
- Workload monitor

Microstores

- Resource manager
- Free resource pool

MOS setup
Initially, the algorithm allocates the same amount of resources to each microstore conservatively then use greedy approach for resource allocation.

- Keep track of free set of resources (including hardware configuration, current load served, and the resource utilization such as CPU and network bandwidth utilization).

- Periodically collect monitoring data from each microstore to aggressively increase and linearly decrease resources from each microstore.
Outline

Introduction
Motivation
Contribution
Design
Evaluation
Preliminary evaluation via simulation – Experimental setup

- **Compute nodes:**
  - 3 – 32 core machines
  - 4 – 16 core
  - 31 – 8 core machines
  - 12 – 4 core machines

- **Network:**
  - 18 – 10 Gbps
  - 32 – 1 Gbps NICs

- HDD to SSD ratio was 70% to 30%.
Aggregated throughput

Small objects

Throughput ($10^3$ QPS)

Time (min)

Default
MOS static
MOS dynamic

Large objects

Throughput (GB/s)

Time (min)

Default
MOS static
MOS dynamic

Small objects

Large objects
Aggregated throughput

Small objects

Large objects

Throughput (GB/s)

Throughput (10^3 QPS)

Time (min)

Default
MOS static
MOS dynamic

Default
MOS static
MOS dynamic

Small objects

Large objects
Aggregated throughput

Small objects

Large objects
Timeline under dynamically changing workloads

Throughput (10^3 QPS) vs Time (min)

Stage 1 | Stage 2 | Stage 3 | Stage 4

Throughput (GB/s)

A | B | C | D
Resource utilization timeline

CPU
Network

Utilization (%)

Time (min)

A
B
C
D
Related Work

- **MET** proposes several system metrics that are critical for a NoSQL database and highly impacts server utilization’s estimation.

- **ϕ Sched** and **Walnut** propose sharing of hardware resources across clouds of different types.

- **CAST** and its extension perform coarse-grained cloud storage (including object stores) management for data analytics workloads.

- **IOFlow** solves a similar problem by providing a queue and control functionality at two OS stages – the storage drivers in the hypervisor and the storage server.
Conclusion

- We performed exhausted study of cloud object stores.
- We proposed a set of rules to help cloud object store administrator to efficiently utilize resources.
- We presented MOS which can outperform extant object store under multi-tenant environment.
- Our analysis shows that it is possible to exploit heterogeneity inherited by modern datacenter to the advantage of object store providers.
http://research.cs.vt.edu/dssl/

Ali Anwar  Yue Cheng  Aayush Gupta  Ali R. Butt