MOS: Taming the Cloud Object Storage

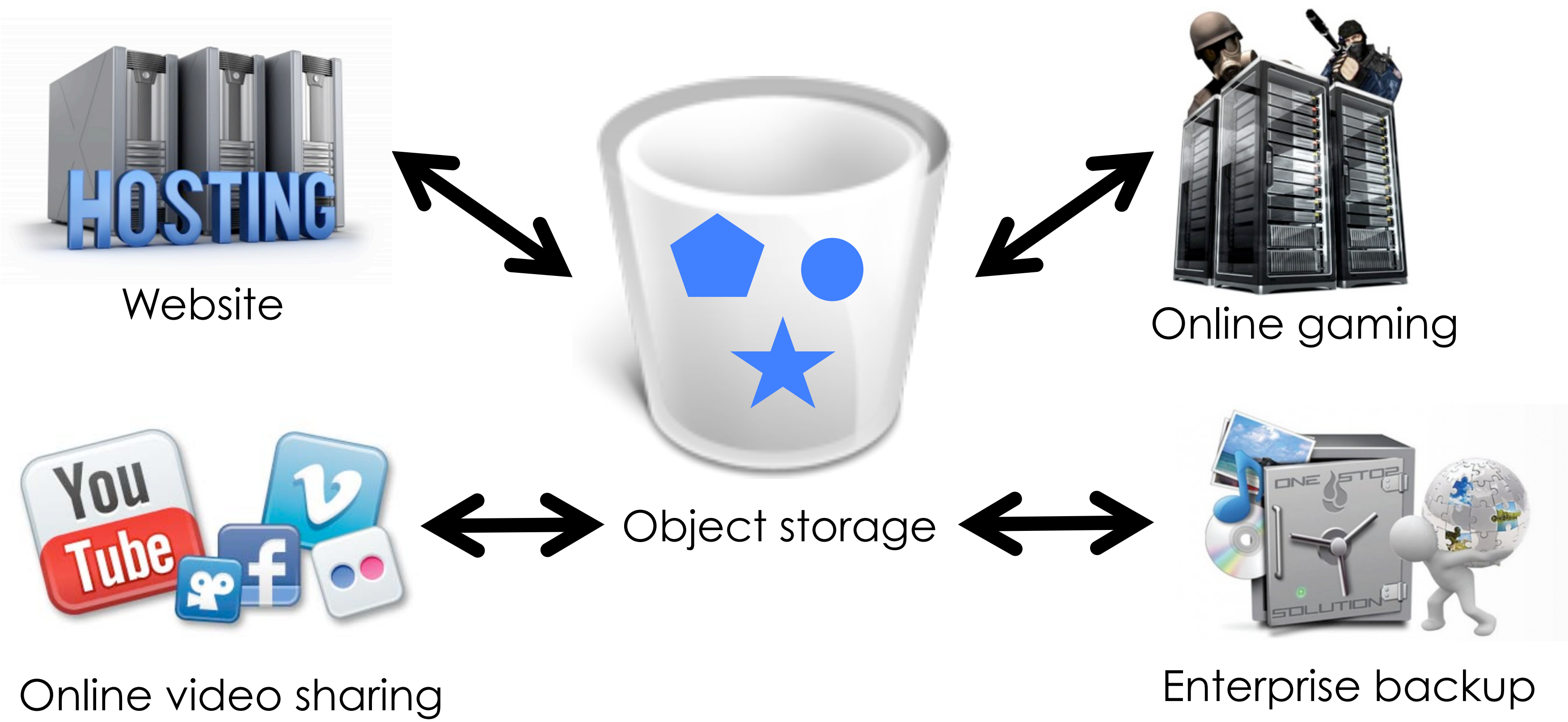
Ali Anwar★, Yue Cheng★, Aayush Gupta†, Ali R. Butt★

★*Virginia Tech* & †*IBM Research – Almaden*

Cloud object stores enable cost-efficient data storage

Object storage

Cloud object store supports various workloads



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

Onlin

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

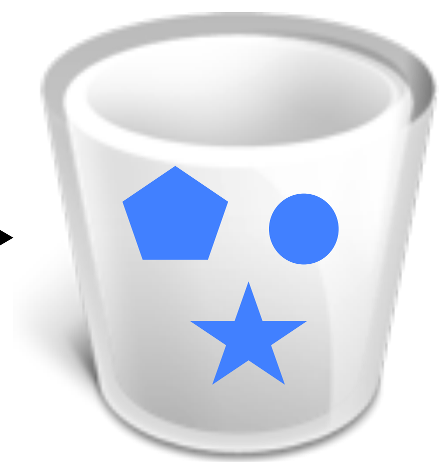
Online video sharing

Enterprise backup

Small objects

Website

Get: 90%, Put: 5%, Delete5:%



Object storage

~ 1-100 KB

Online gaming

Get: 5%, Put: 90%, Delete5:%

Large objects

Online video sharing

Get: 90%, Put: 5%, Delete5:%

Object storage

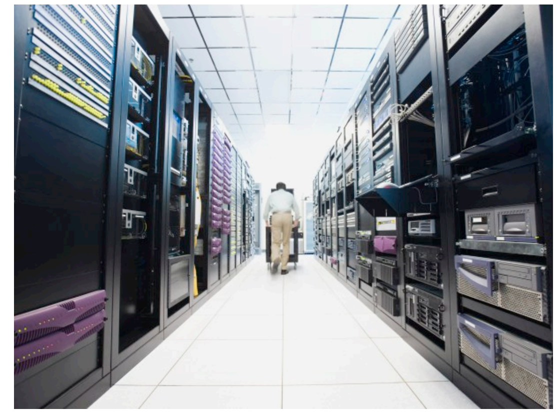
~ 1-100 MB

Enterprise backup

Get: 5%, Put: 90%, Delete5:%

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

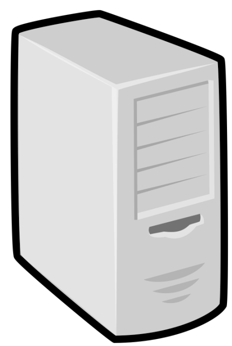
Design

Evaluation

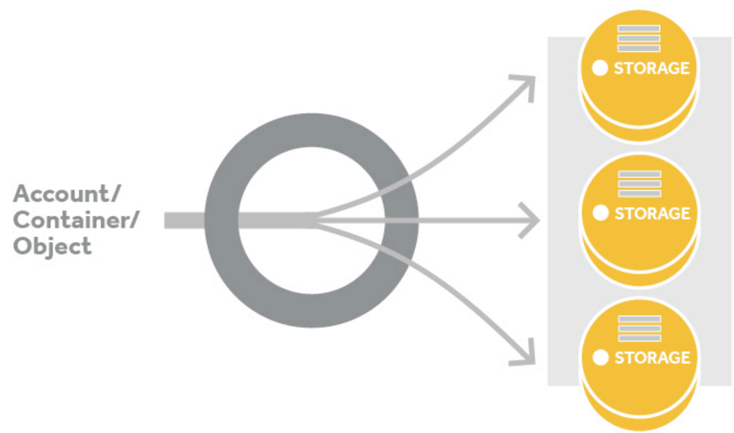
Background: Swift object store

Object storage

=

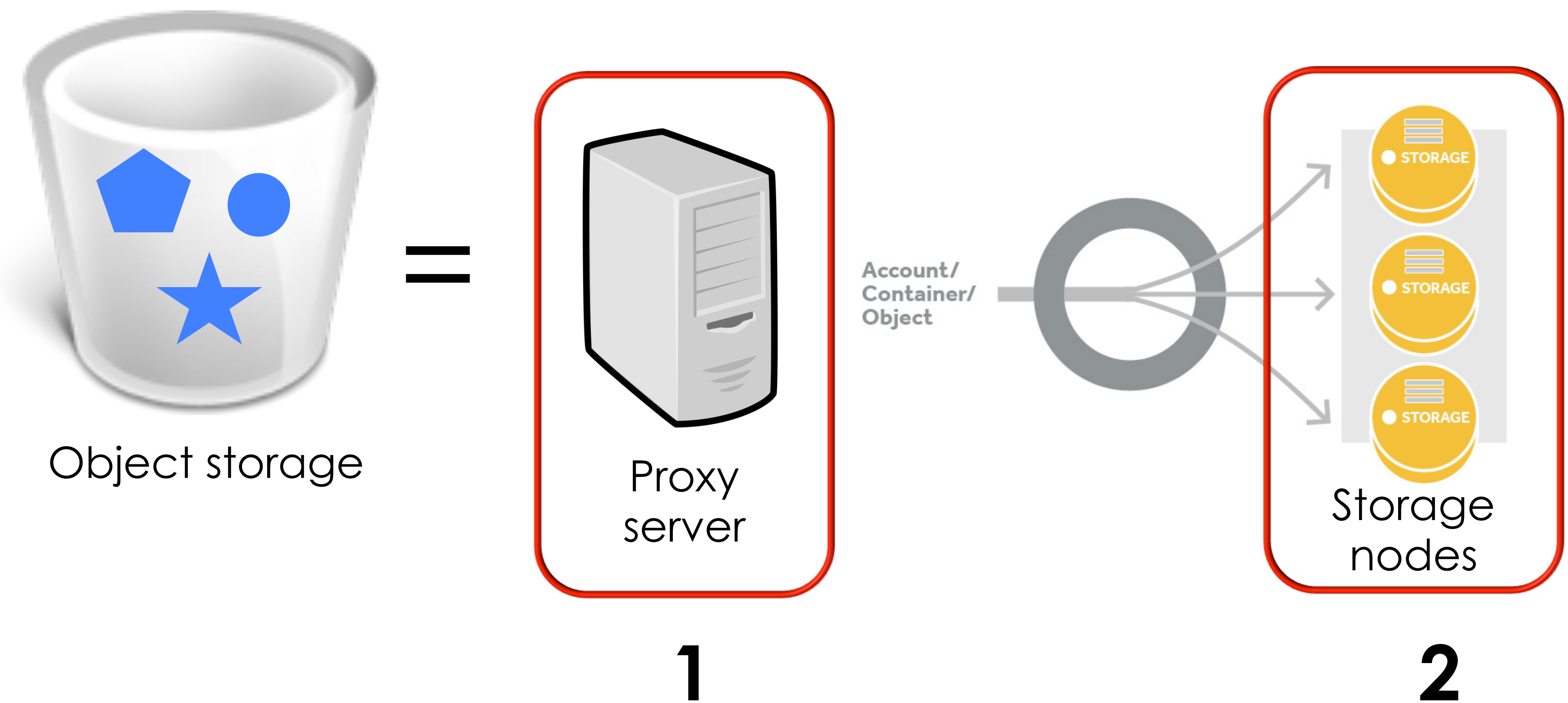


Proxy server

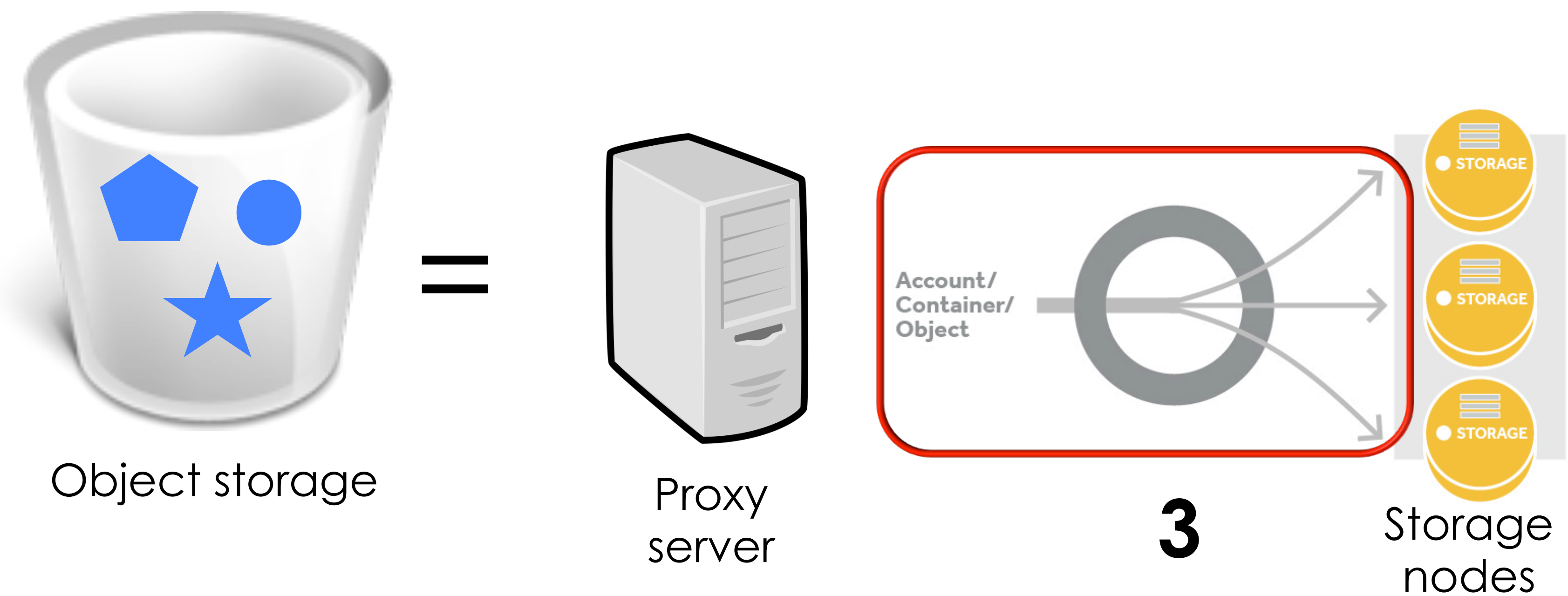


Storage nodes

Swift: Proxy and Storage servers



Swift: Ring architecture



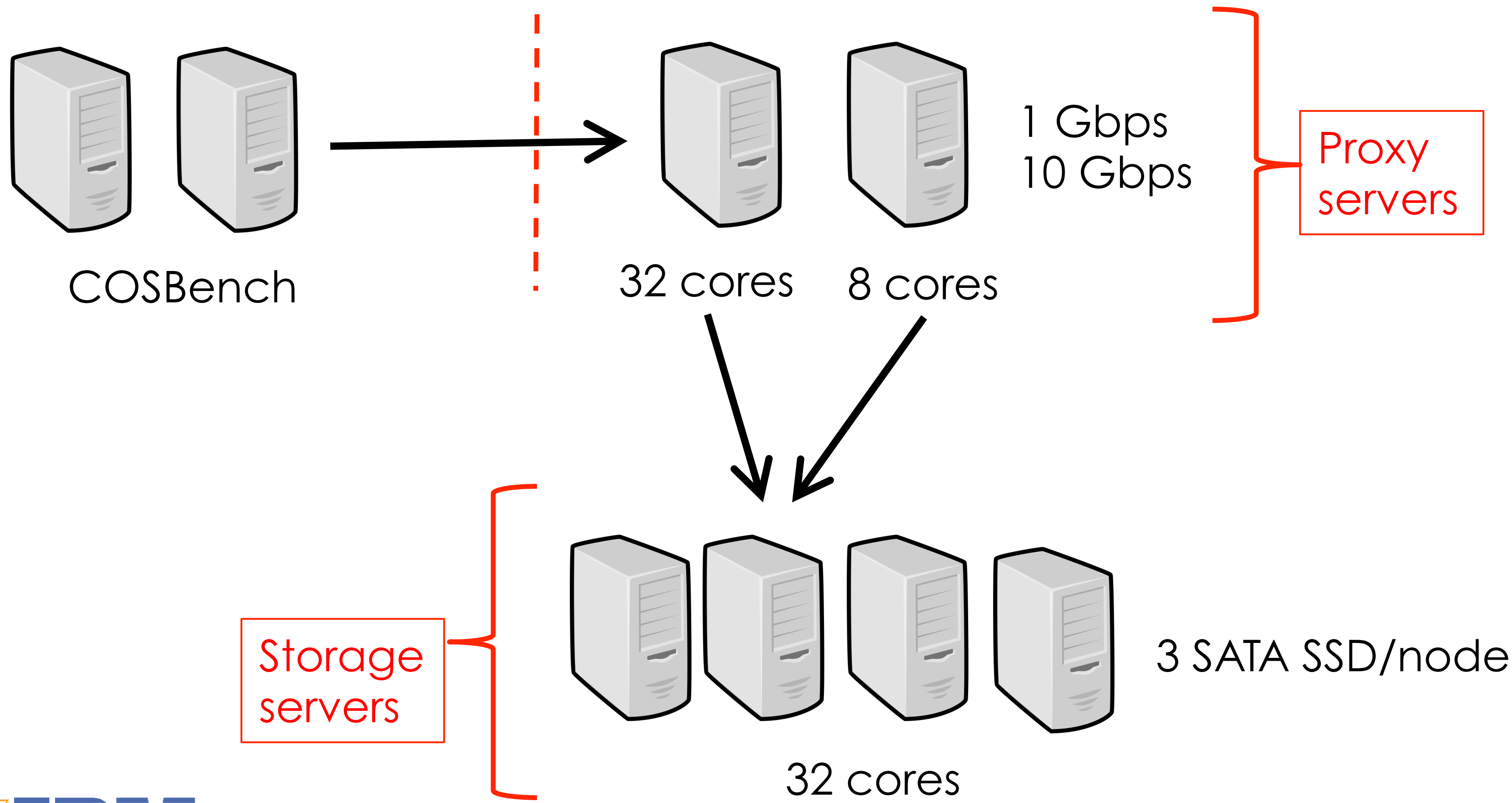
Benchmark used: CosBench

- COSBench is Intel developed **Bench**mark to measure **C**loud **O**bject **S**torage 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

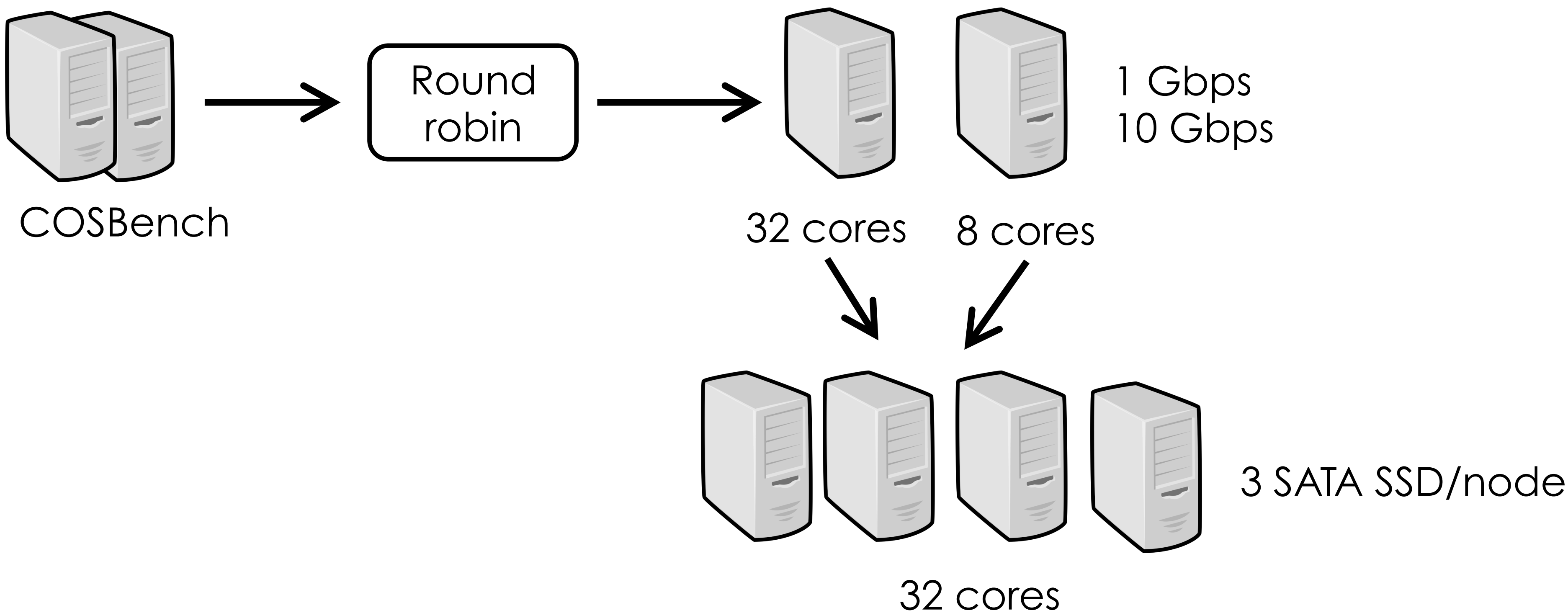
Workload used

Workload	Workload Characteristics		Application scenario
	Object Size	Distribution	
Workload A	1 – 128 KB	**G: 90%**, P: 5%, D:5%	Web hosting
Workload B	1 – 128 KB	G: 5%, **P: 90%**, D:5%	Online game hosting
Workload C	1 – 128 MB	**G: 90%**, P: 5%, D:5%	Online video sharing
Workload D	1 – 128 MB	G: 5%, **P: 90%**, D:5%	Enterprise backup

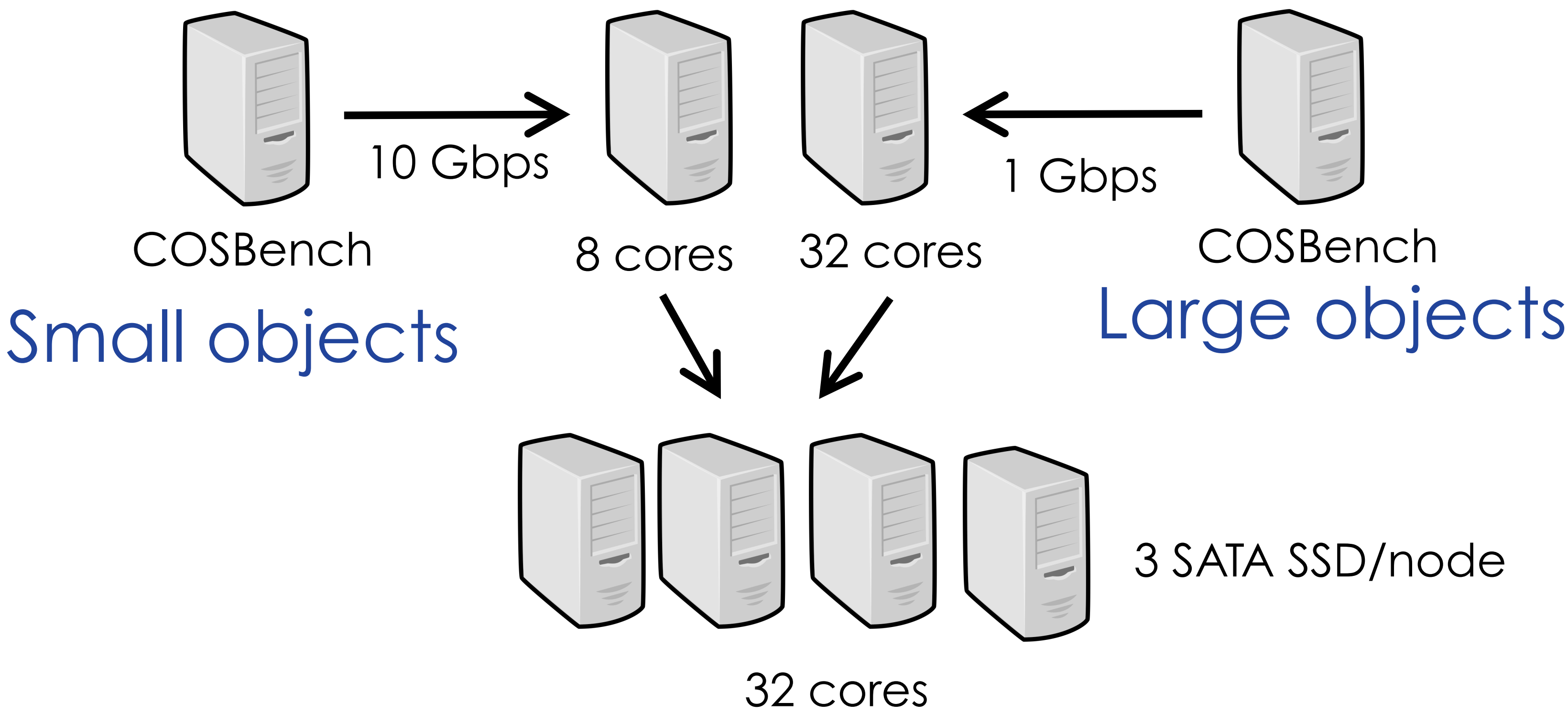
Experimental setup for motivational study



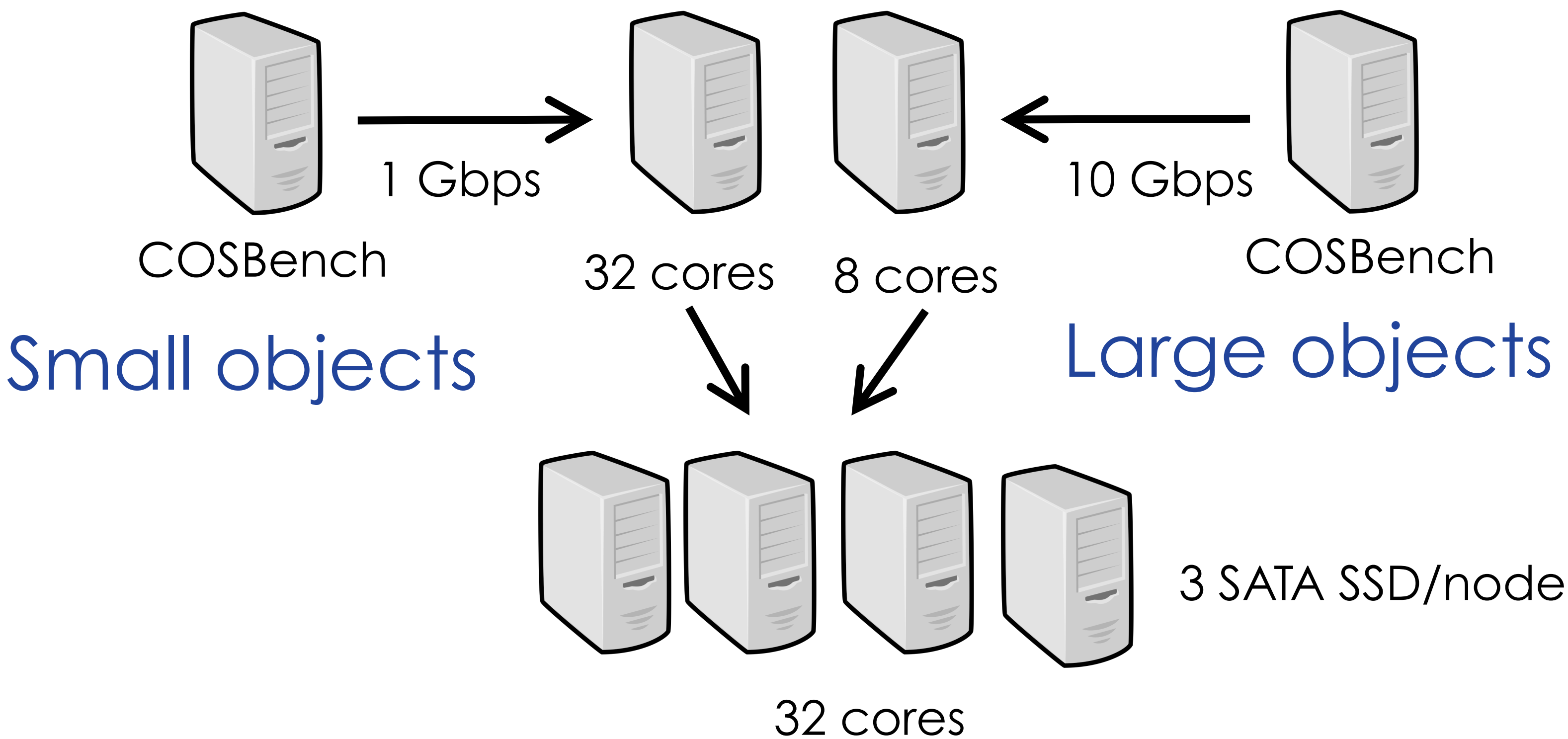
Configuration 1 – Default monolithic



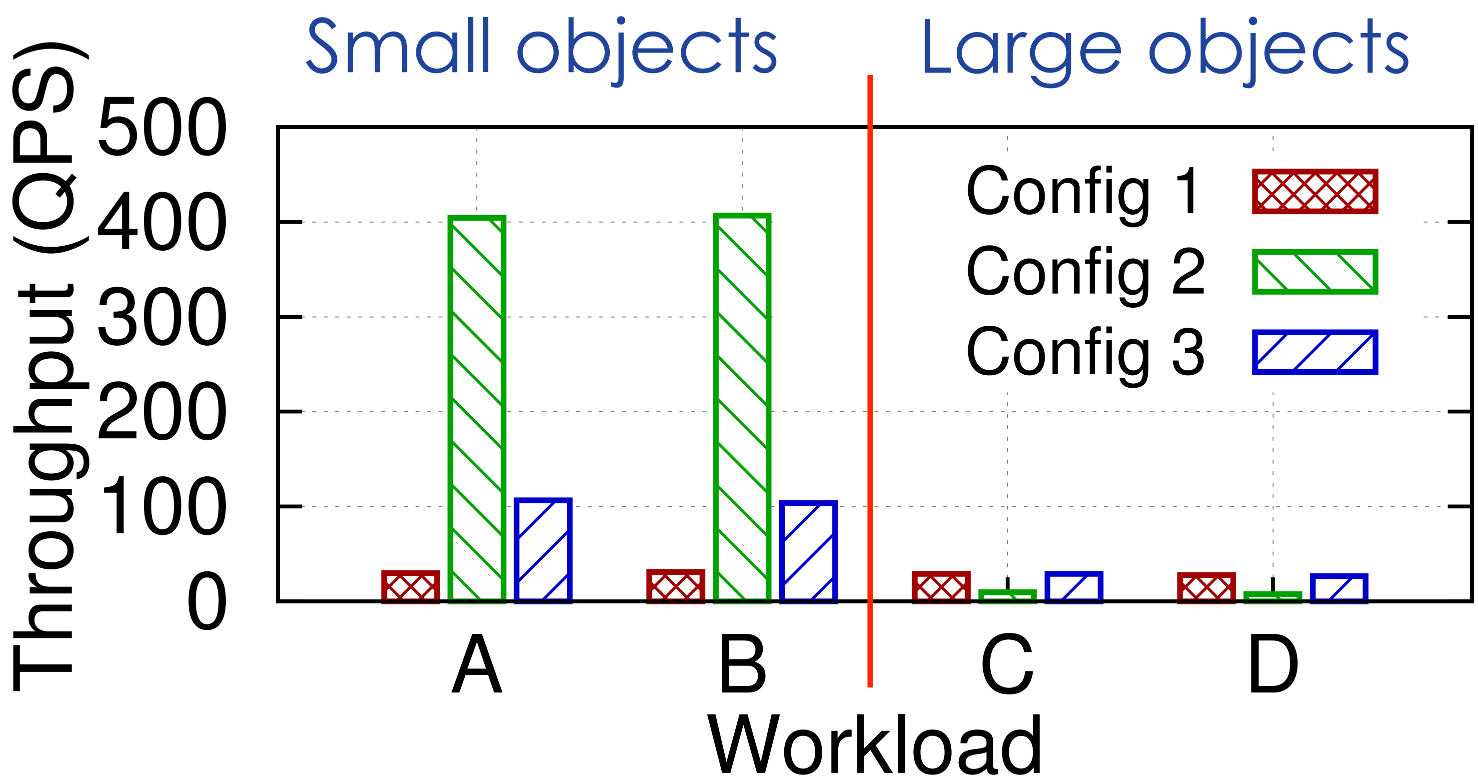
Configuration 2 – Favors small objects



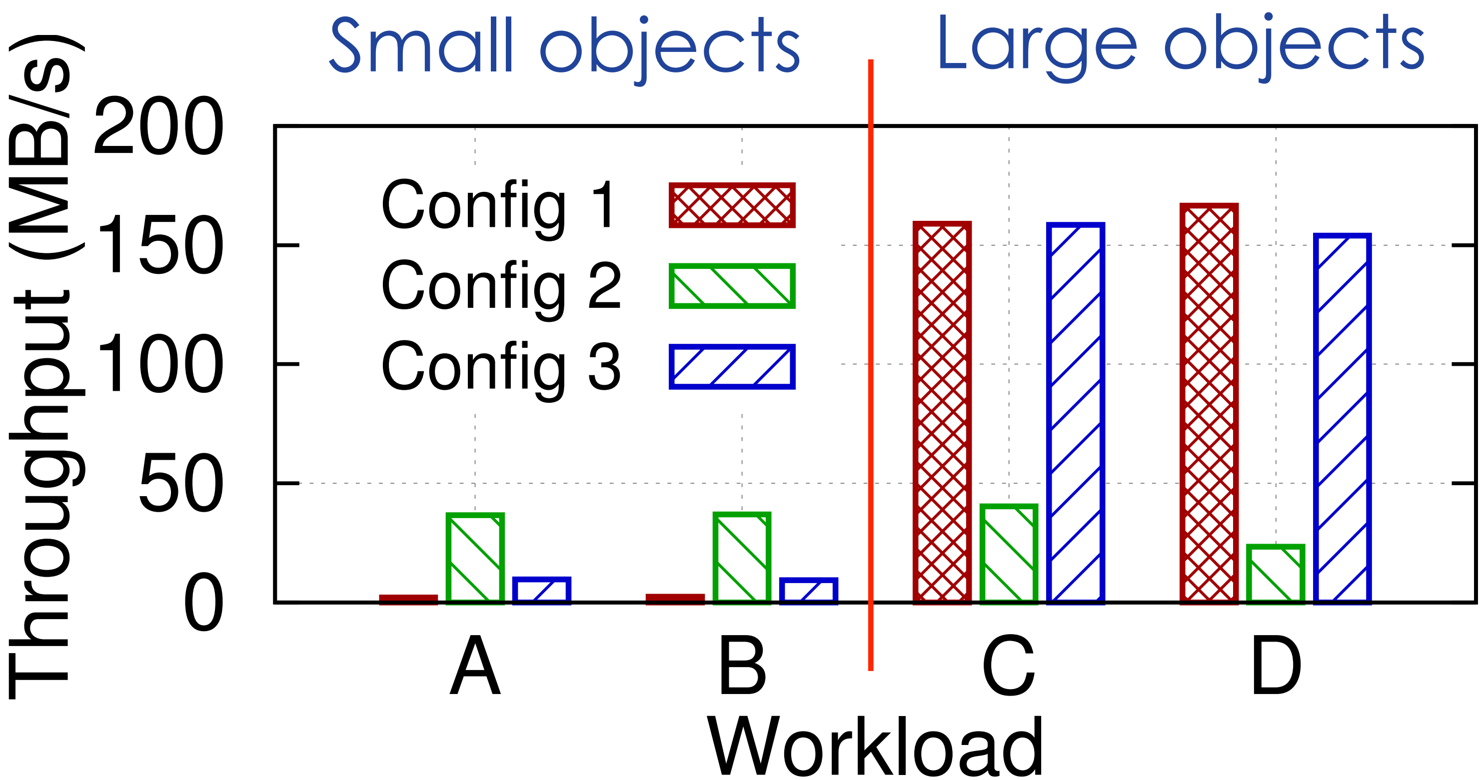
Configuration 3 – Favors large objects



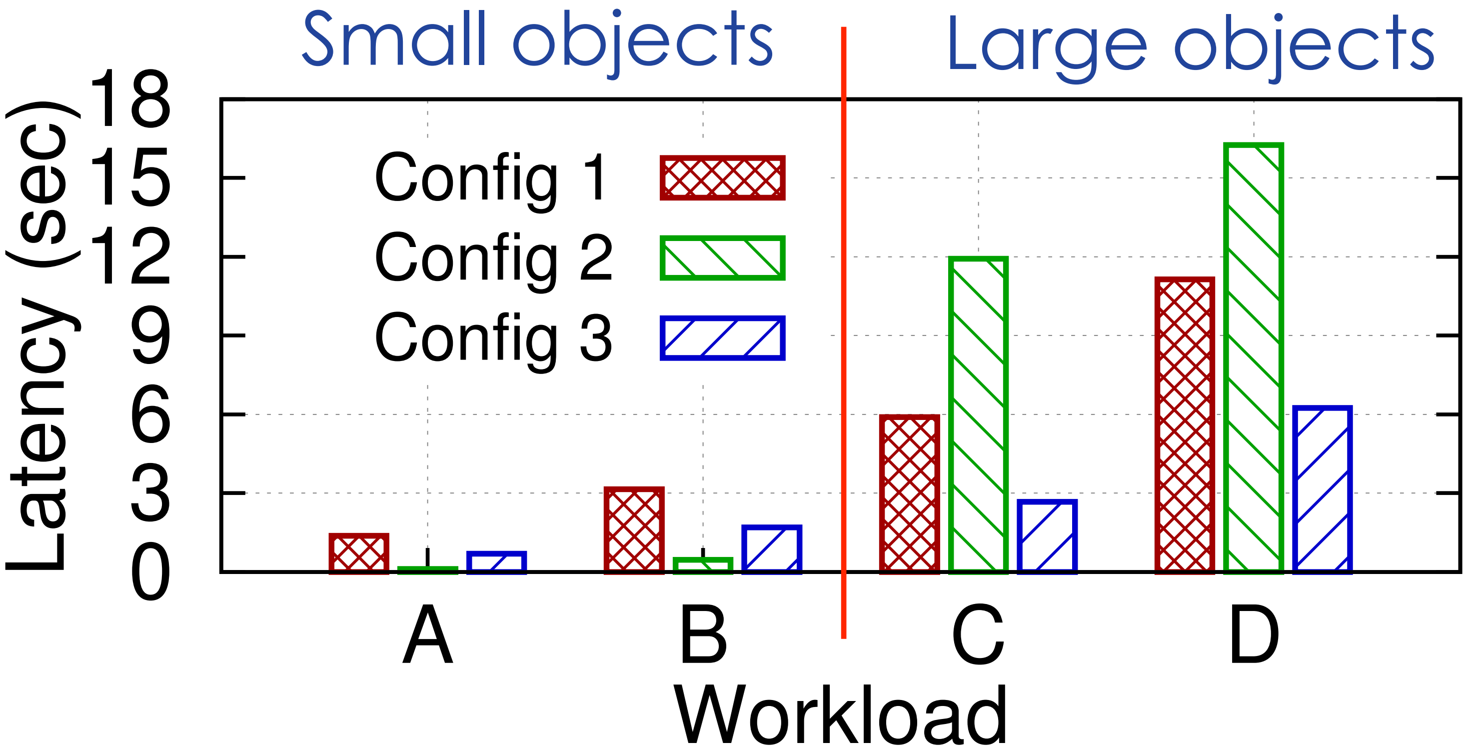
Performance under multi tenant environment – Workload A & B



Performance under multi tenant environment- – Workload A & B



Performance under multi tenant environment - latency



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, **M**icro **O**bject **S**torage:
  - 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~~

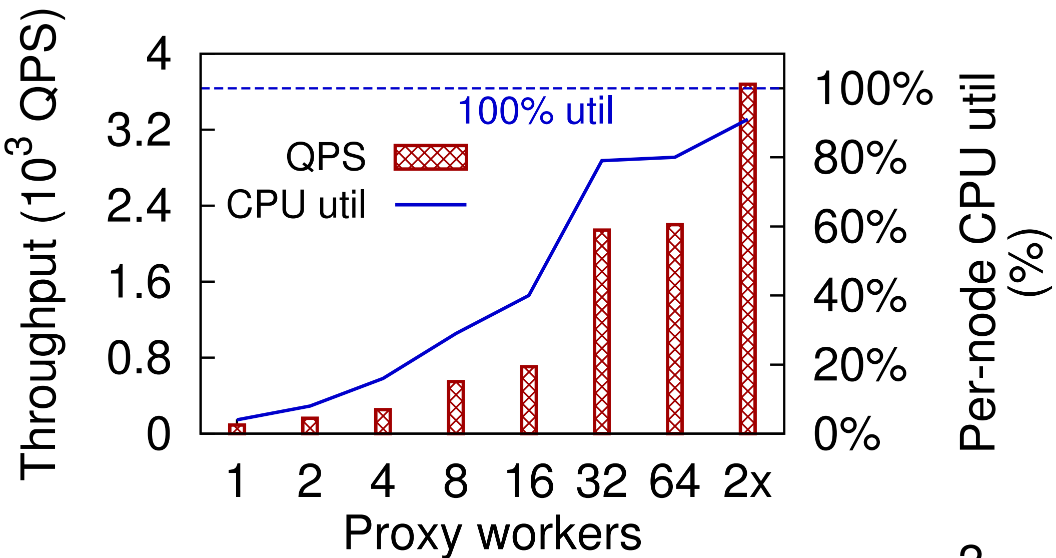
Design

Evaluation

Design criteria for MOS

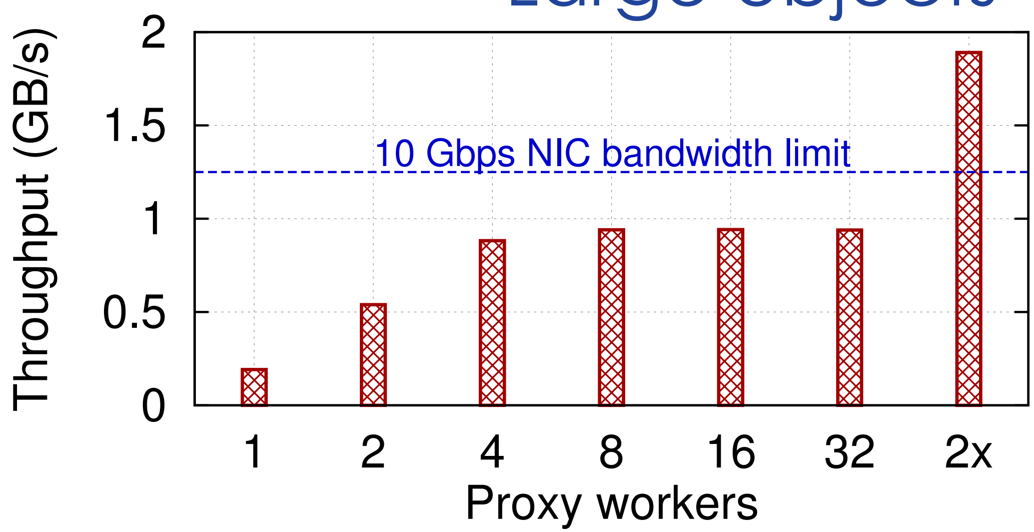
- We studied the effect of three knobs on performance of a typical object store to come up with design rules/ rules of thumb
  - Proxy Server settings
  - Storage Server settings
  - Hardware changes

Effect of Proxy server settings

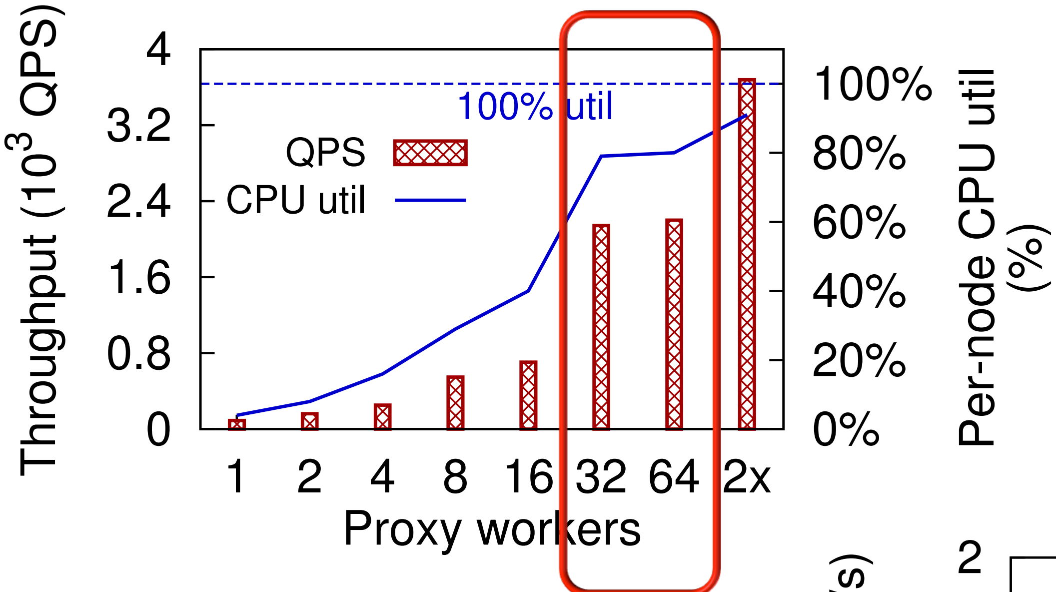


Small objects

Large objects

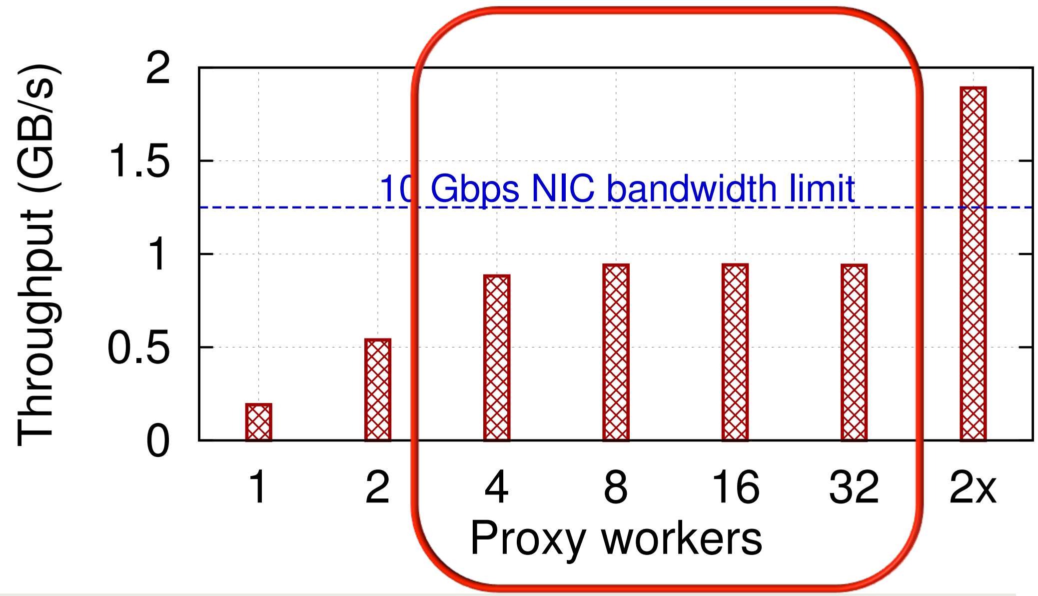


Effect of Proxy server settings

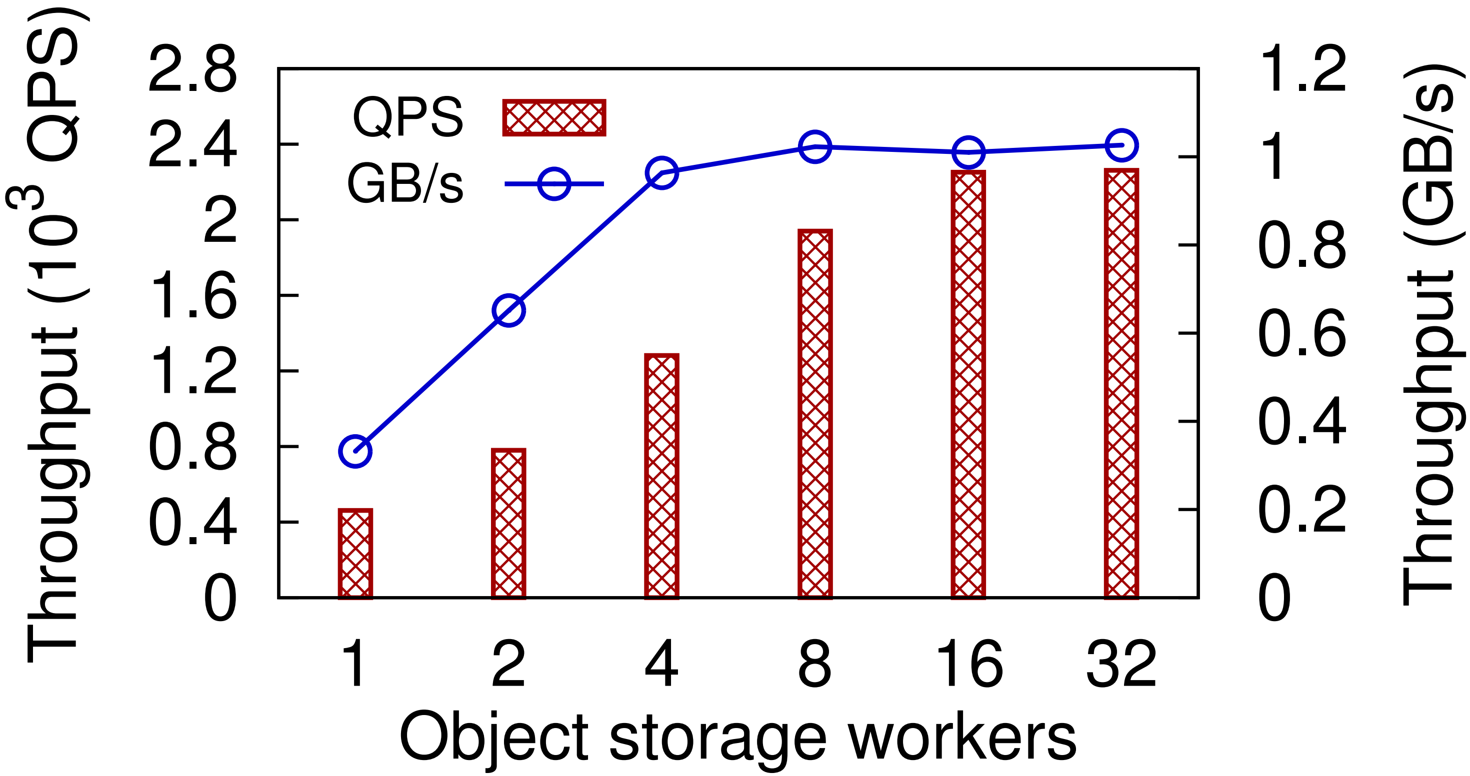


Small objects

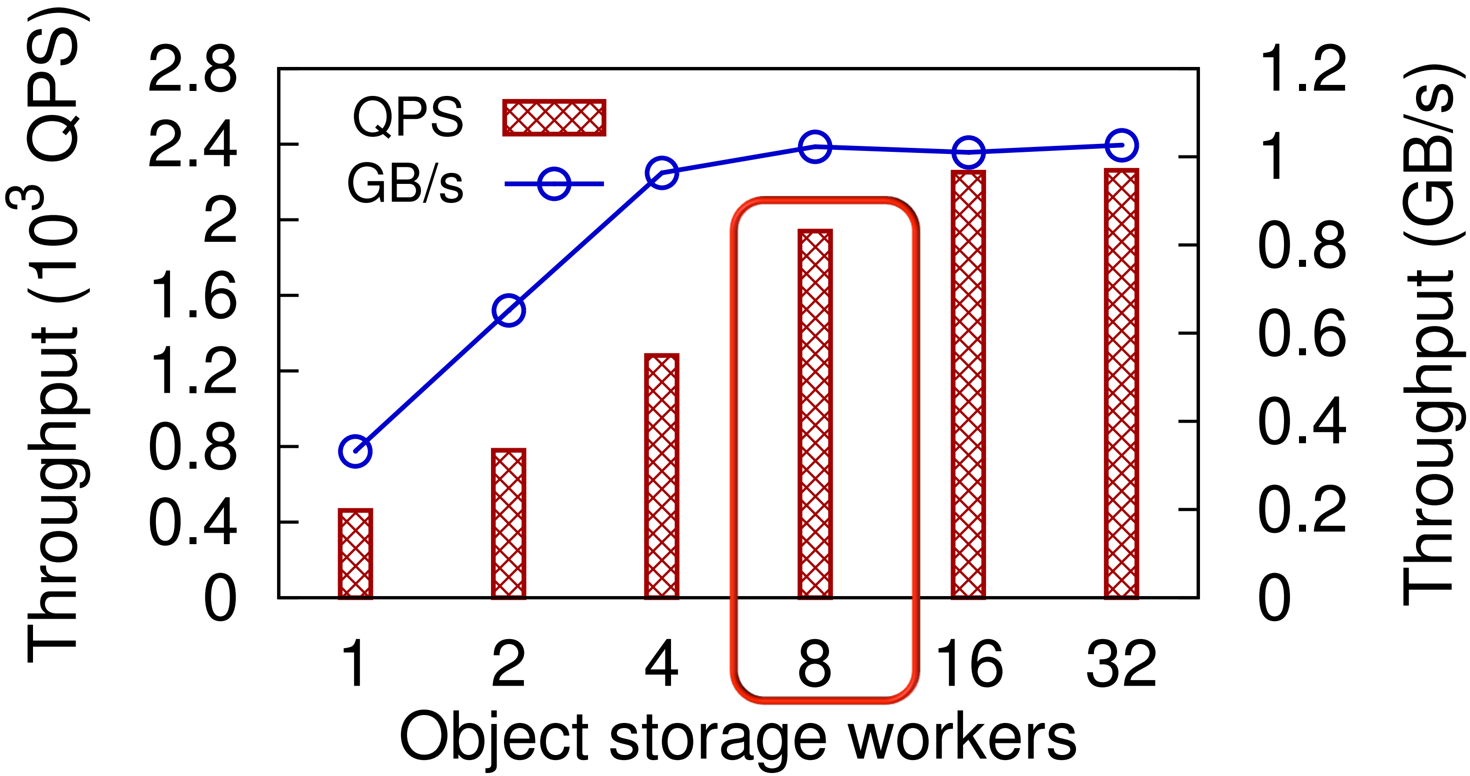
Large objects



Effect of Storage server settings

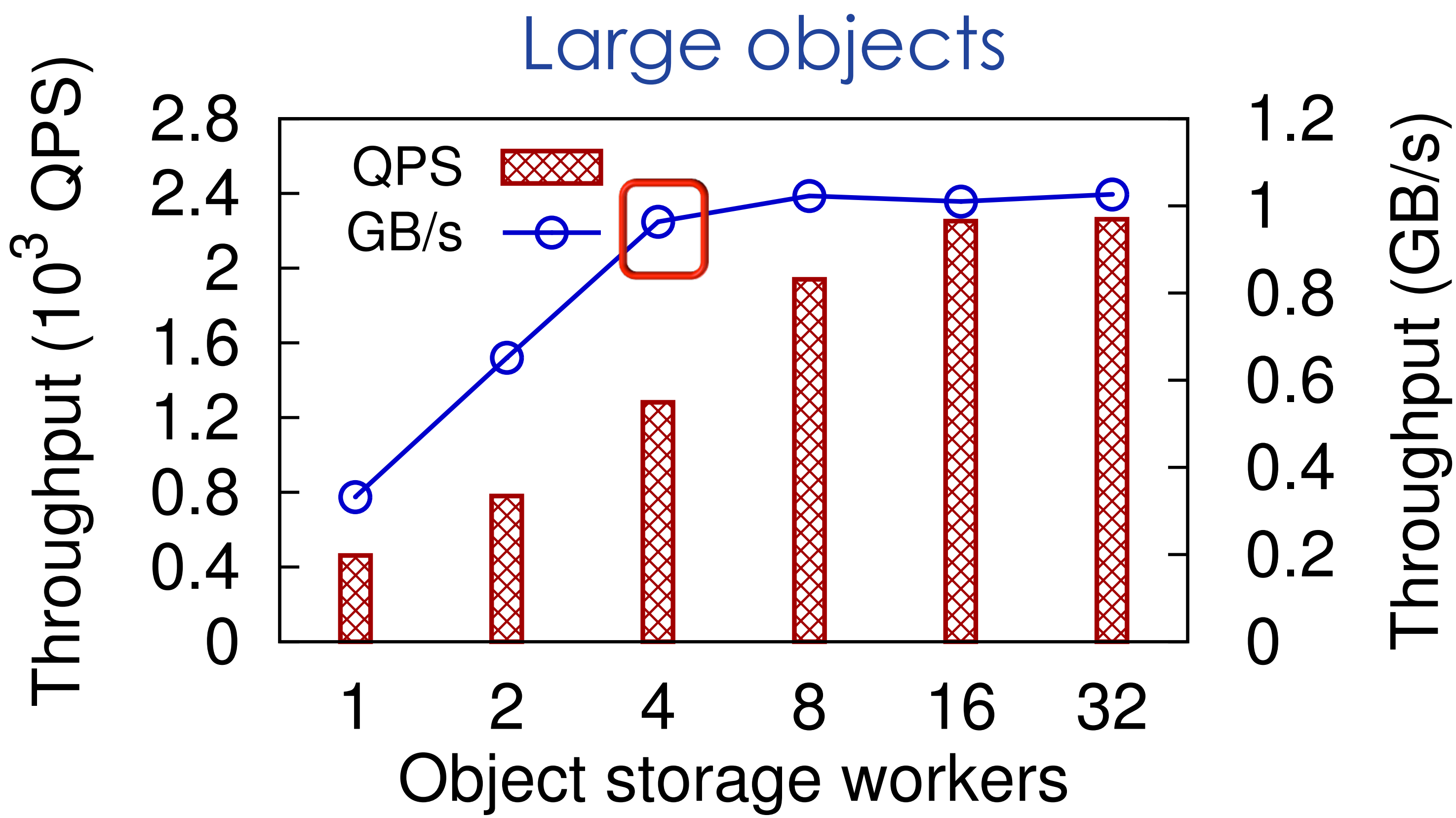


Effect of Storage server settings



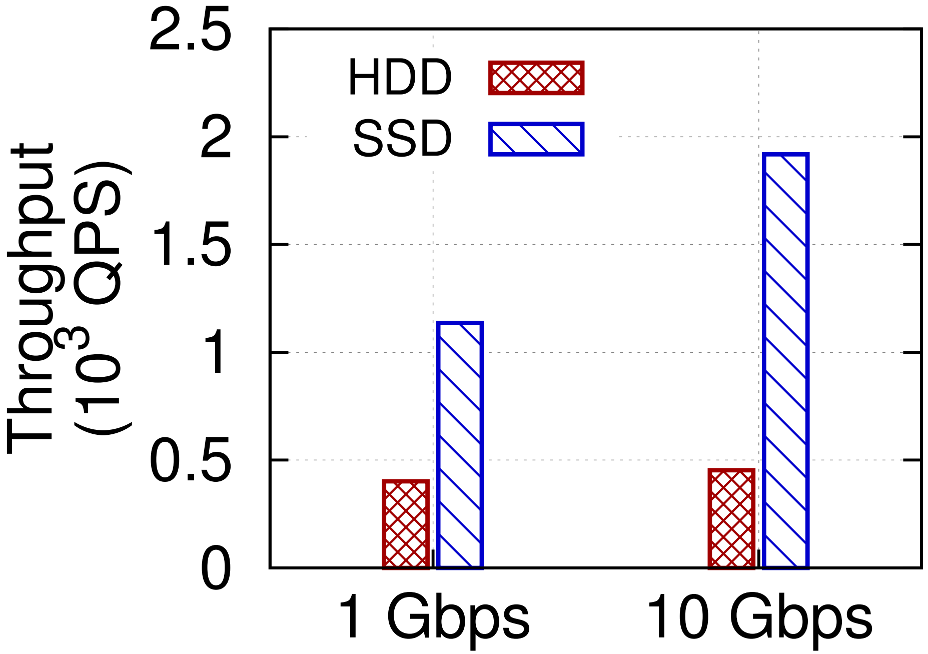
Small objects

Effect of Storage server settings

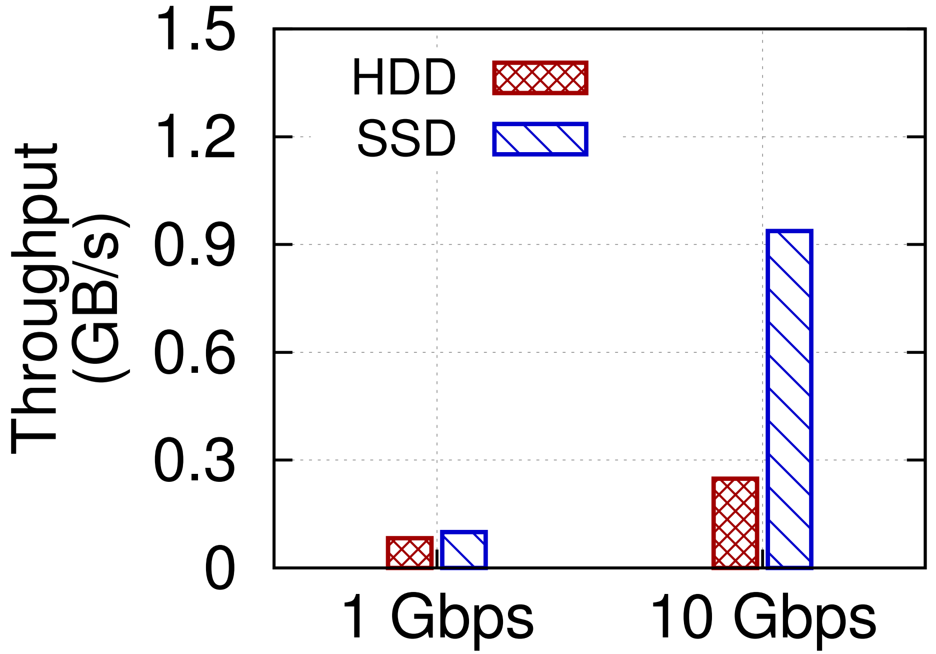


Effect of hardware settings

Small objects



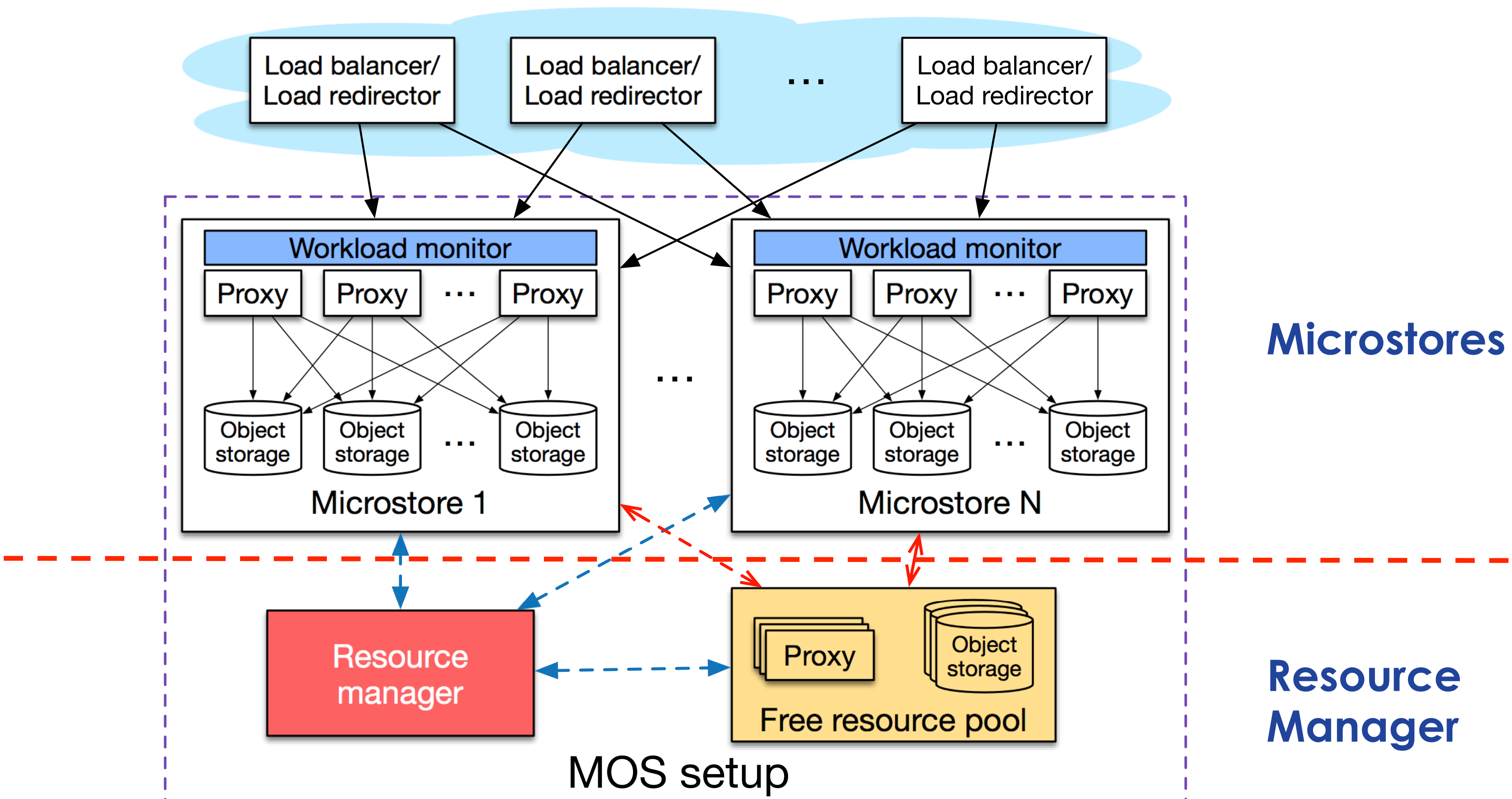
Large objects



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
- $proxyCores = storageNodes * coresPerStorageNode$
- $BW_{proxies} = storageNodes * BW_{storageNode}$
- 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
Load balancer/
Load redirector
...
Load balancer/
Load redirector
Workload monitor
Proxy
Proxy
...
Proxy
Object storage
Object storage
...
Object storage
Microstore 1
...
Workload monitor
Proxy
Proxy
...
Proxy
Object storage
Object storage
...
Object storage
Microstore N
Resource manager
Proxy
Object storage
Free resource pool
MOS setup
Microstores
Resource Manager

Resource Provisioning Algorithm

- 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~~

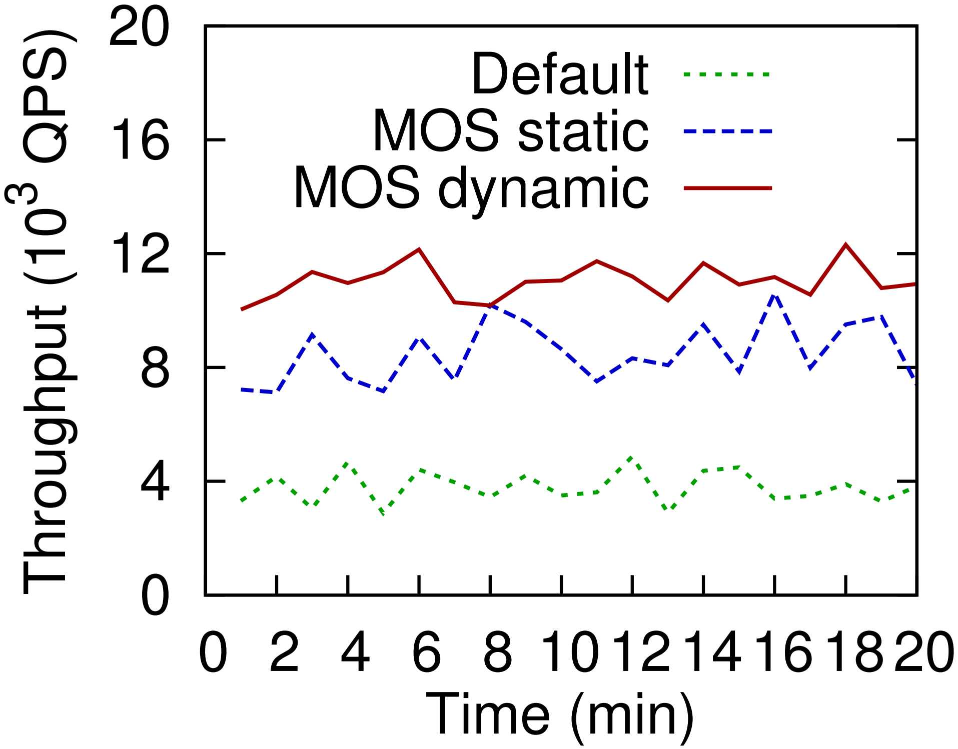
Evaluation

Preliminary evaluation via simulation – Experimental setup

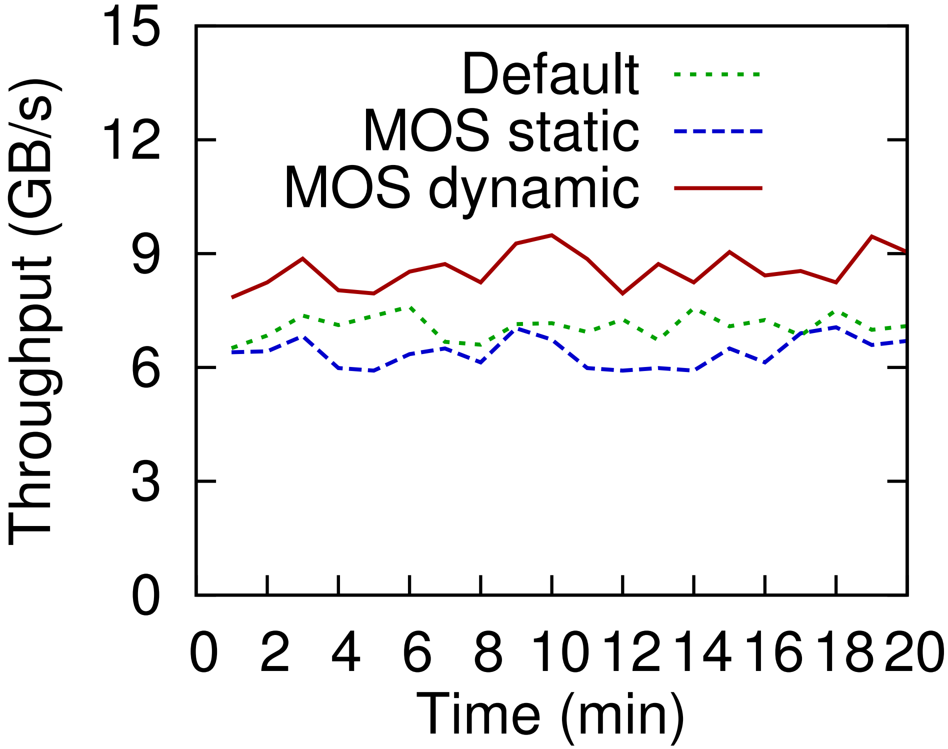
- Compute nodes:
  - **3** – 32 core machines
  - **4** – 16 core
  - **31** – 8 core machines
  - **12** – 4 core machines
- Network:
  - **18** – 10 Gbps
  - **32** – 1 Gbps NICs
- HDD to SSD ratio was 70% to 30%.

Aggregated throughput

Small objects

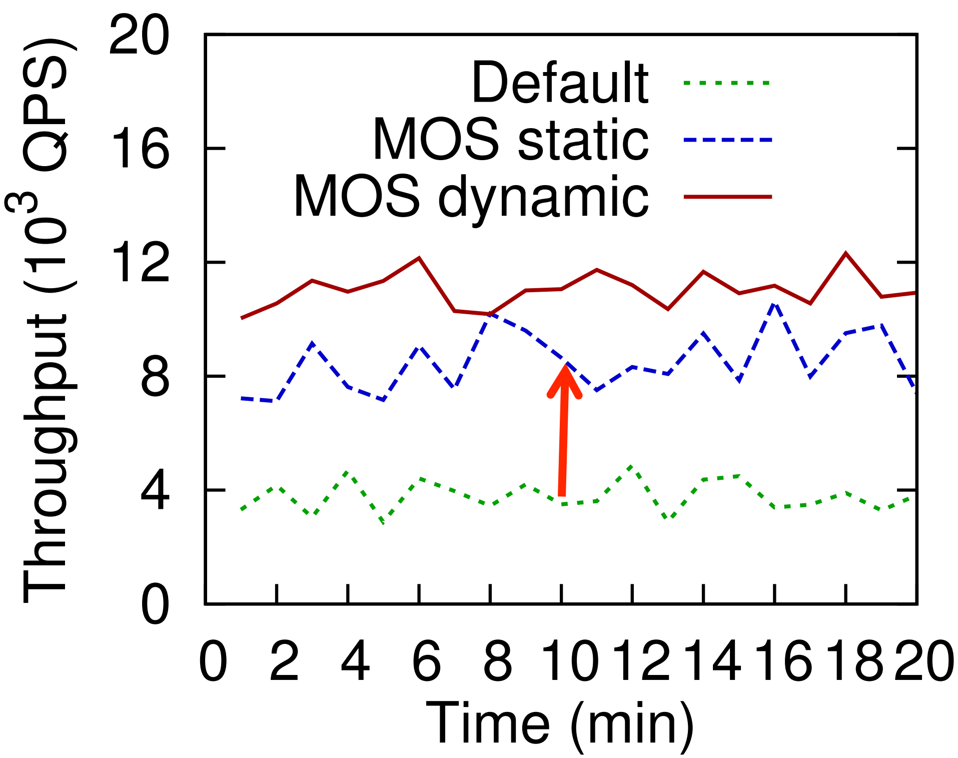


Large objects

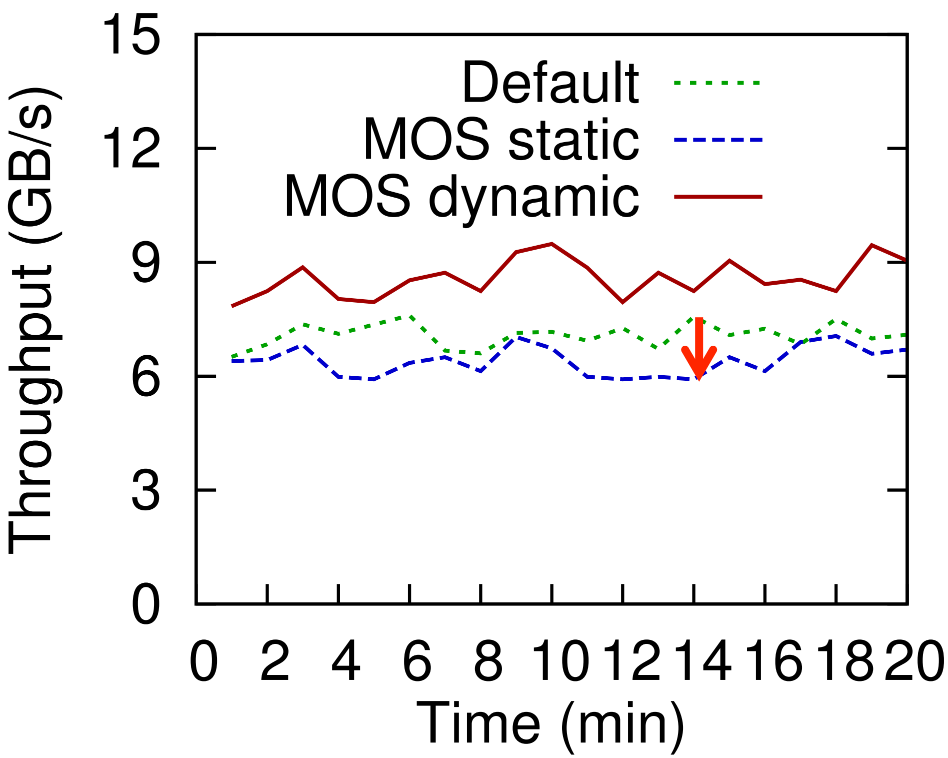


Aggregated throughput

Small objects

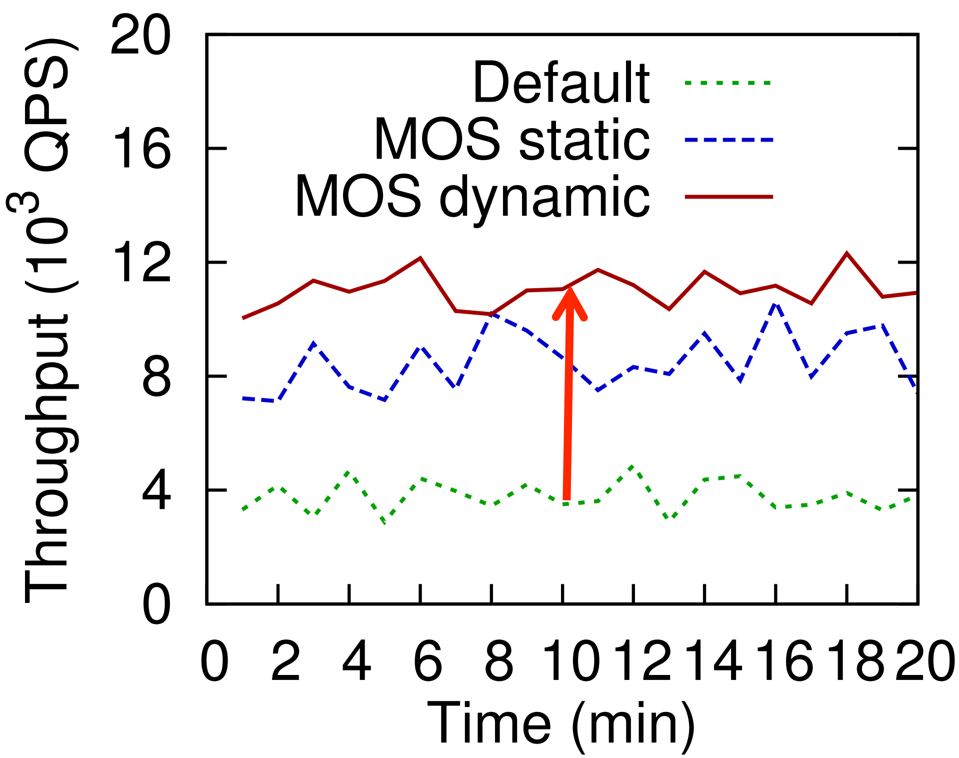


Large objects

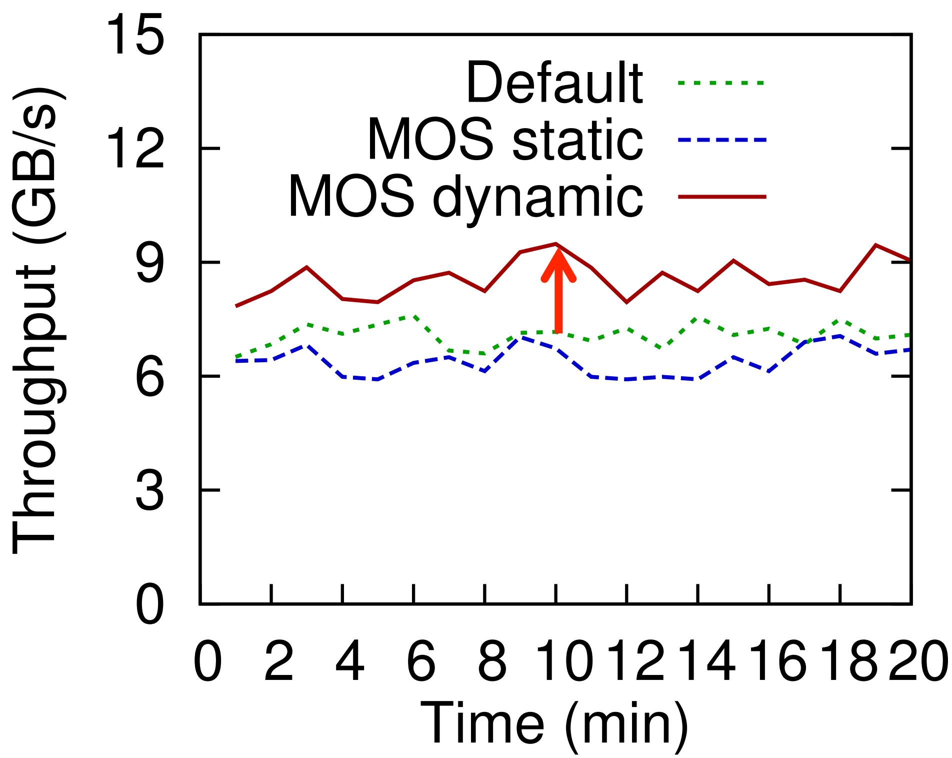


Aggregated throughput

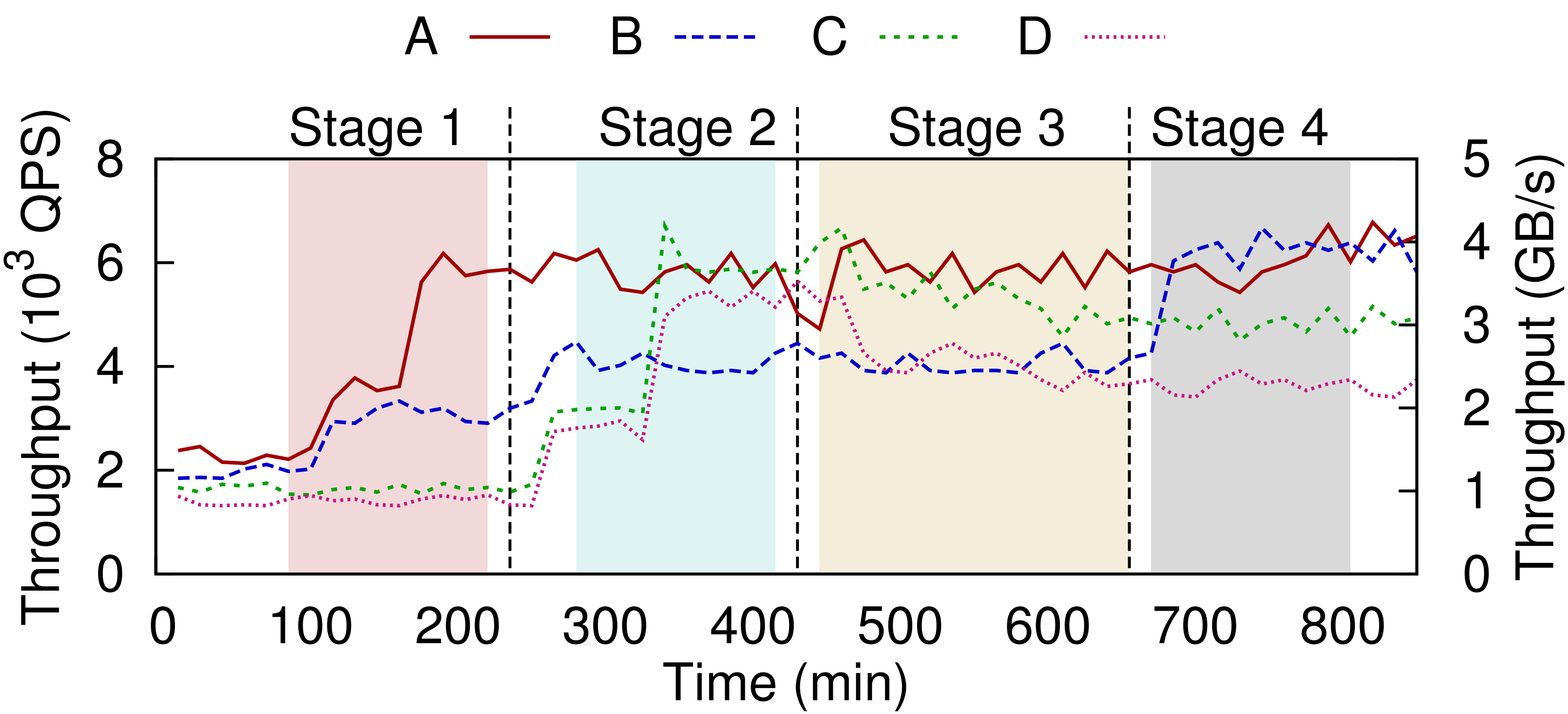
Small objects



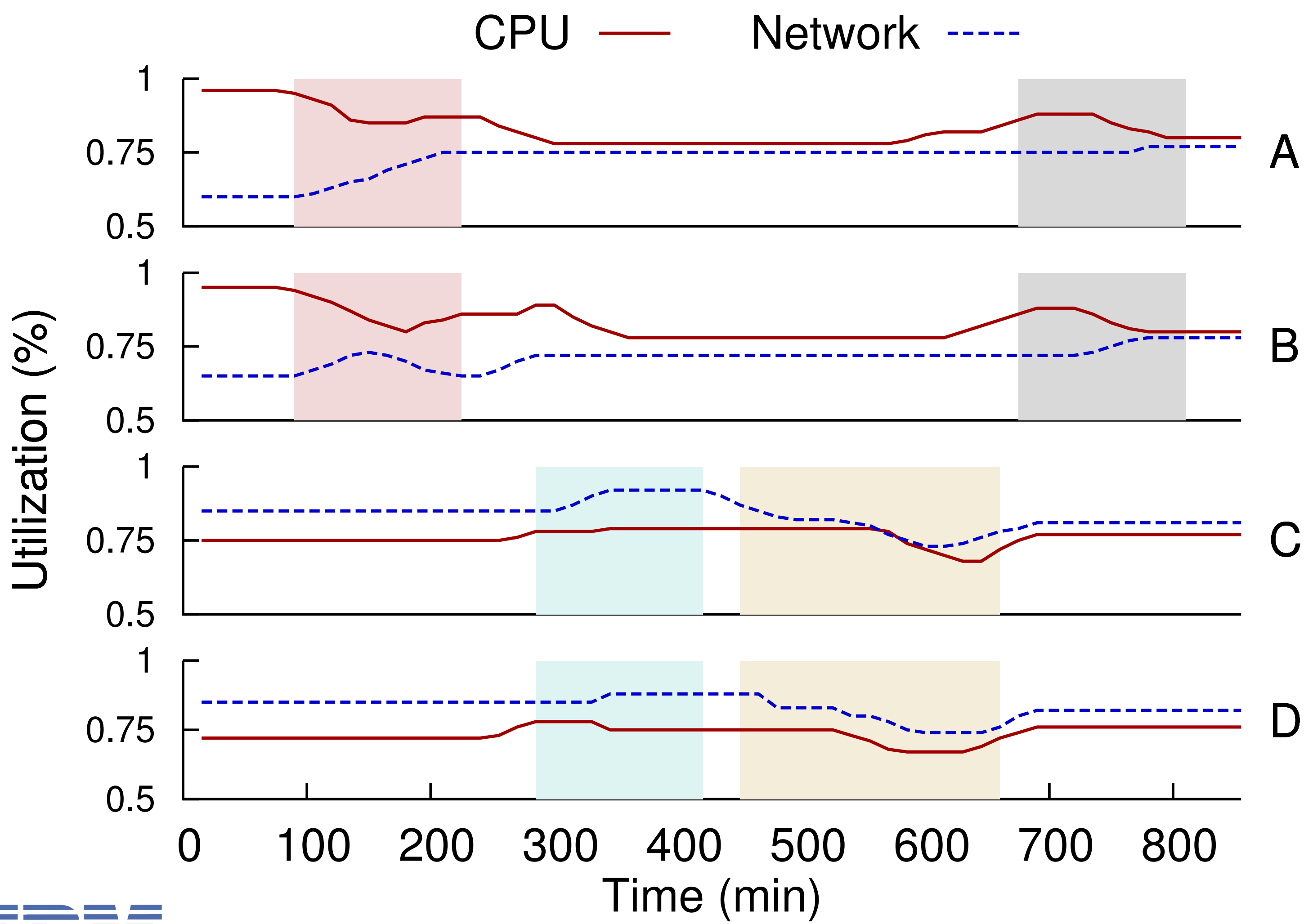
Large objects



Timeline under dynamically changing workloads



Resource utilization timeline



Related Work

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

Q&A

MOS

Thank you!

http://research.cs.vt.edu/dssl/

Ali Anwar   Yue Cheng   Aayush Gupta   Ali R. Butt