
COSBench
A Benchmark Tool for Cloud Object Storage Services
Qing Zheng, Chen Haopeng
Yaguang Wang, Jiangang Duan, Zhiteng Huang
Shanghai Jiao Tong University, Intel Asia-Pacific R&D Ltd.

Object Storage

Structured data

- profiles/states/transactions/orders

Unstructured data

- photos/documents/movies/backups/logs/archives

Object Storage Model

Object

- object = data + metadata

RESTful interface

- head/get/put/post/delete

Implementations

What's next?

To further drive optimization and innovation

- Understand and compare different cloud object storage services
- locate bottlenecks and improve architectures/algorithms/configurations
- find future research opportunities for both HW solutions and SW solutions

None of this can be achieved without a dedicated benchmark tool for cloud object storage.

COSBench

Major requirements

- scalable
  - support single client load testing, and distributed load testing
- flexible
  - workload model with multiple configuration options
- simple
  - easy to use, real-time performance showing, automatic report generation

Workload Model

Workload

- comprises a number of workers performing a certain set of operations

Access pattern

- operation-percentage/object-range/object-size

Concurrency pattern

- worker-number/container-number

Workload Configuration

- Basic Mode

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="Core"
          description="swift performance test"
          target="Mock"
          credentials="conf/cloudfiles.properties"
          workerCount="4"
          containerCount="8"
          operationCount="0"
          initObjectCount="20"
          runtime="20"
          objectSize="64k"
          bufferSize="128k"
          isPrepared="true"
          readPercentage="100"
          writePercentage="0"
          showStatus="true"
          sampleInterval="5000"
          statusInterval="5000"
          opLogging="false"
          verbose="false"

</workload>
```

Workload Configuration

- Advanced Mode

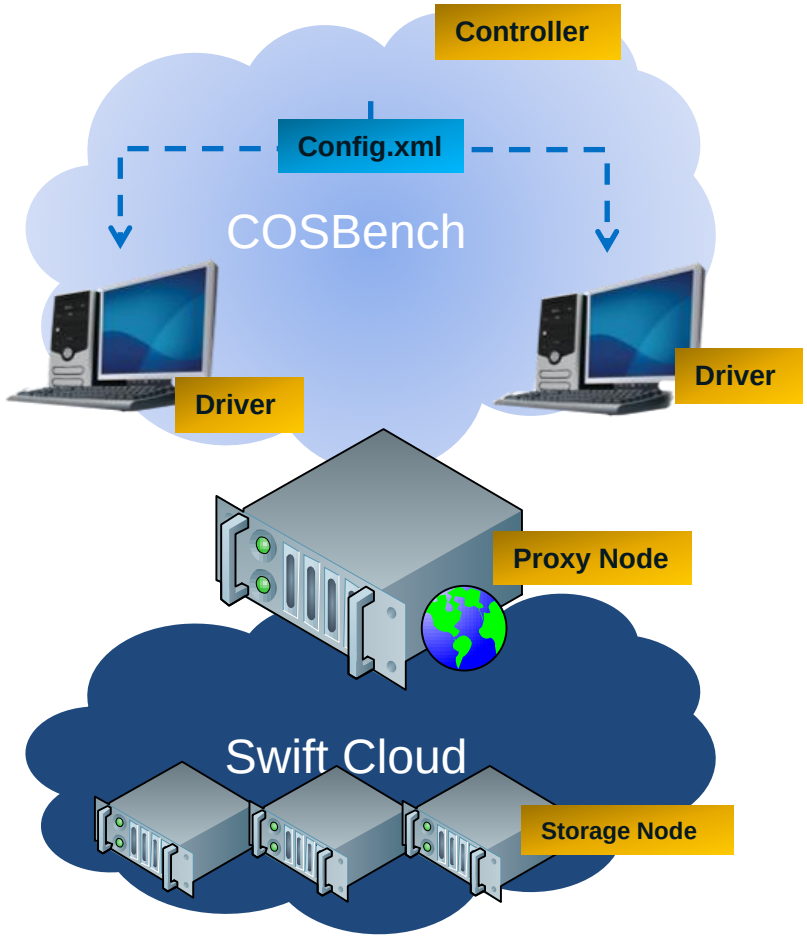
```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="Core" description="swift test" target="Swift"
credentials="conf/cloudfiles.properties" workerCount="2" containerCount="20"
operationCount="0" initObjectCount="10000" runtime="60" isPrepared="true">

  <spec name="64k;50%Read">
    <operation name="READ" percentage="50" objectSize="64k" />
    <operation name="WRITE" percentage="50" objectSize="64k" />
  </spec>

  <spec name="64k;100%Read">
    <operation name="READ" percentage="100" objectSize="64k" />
  </spec>

  <worker name="1" spec="64k;50%Read" containers="1-20" objects="1-5000" />
  <worker name="2" spec="64k;100%Read" containers="1-20" objects="5001-10000" />

</workload>
```

Architecture

Controller
- control all drivers and collect stats

Driver
- driver can run tests w/o controller
- one daemon thread which keeps listening for instructions from controller, and passing log back
- multiple agents which generate load according to workload configuration, the work for each agent can be different

Controller interacts with daemon through
- sending command to control agents
- receiving real-time logs



Metrics

Throughput:

- number f operations completed in one second (op/s)

Response time

- duration between operation initiation and completion (ms)

Bandwidth

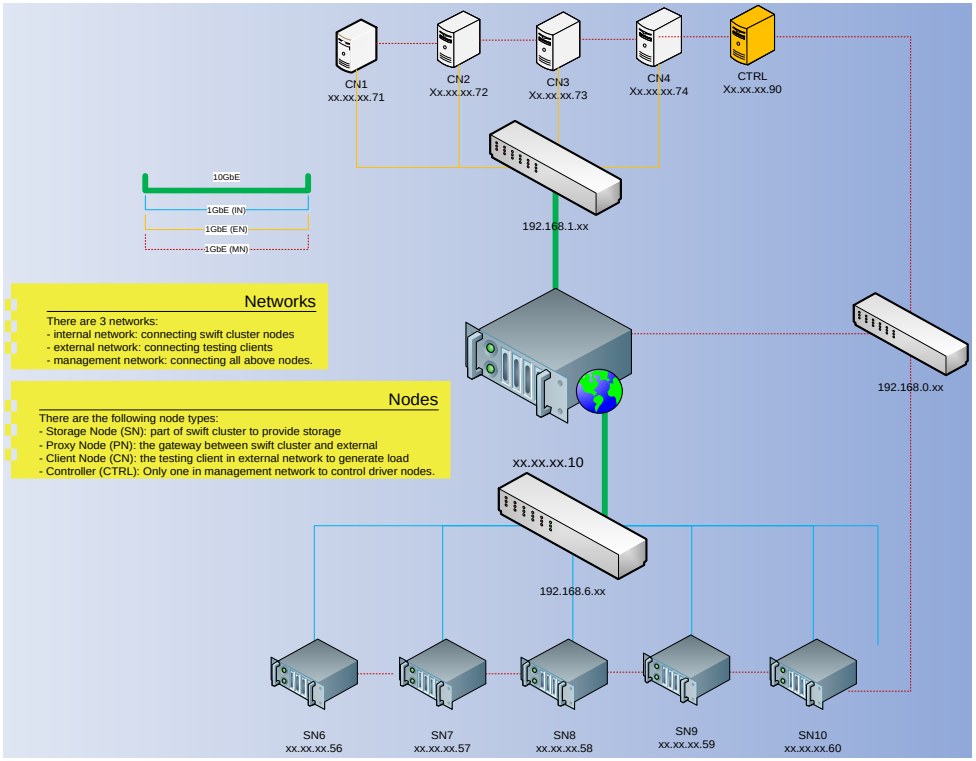
- amount of data transferred in one second (KiB/s)

Success Ratio

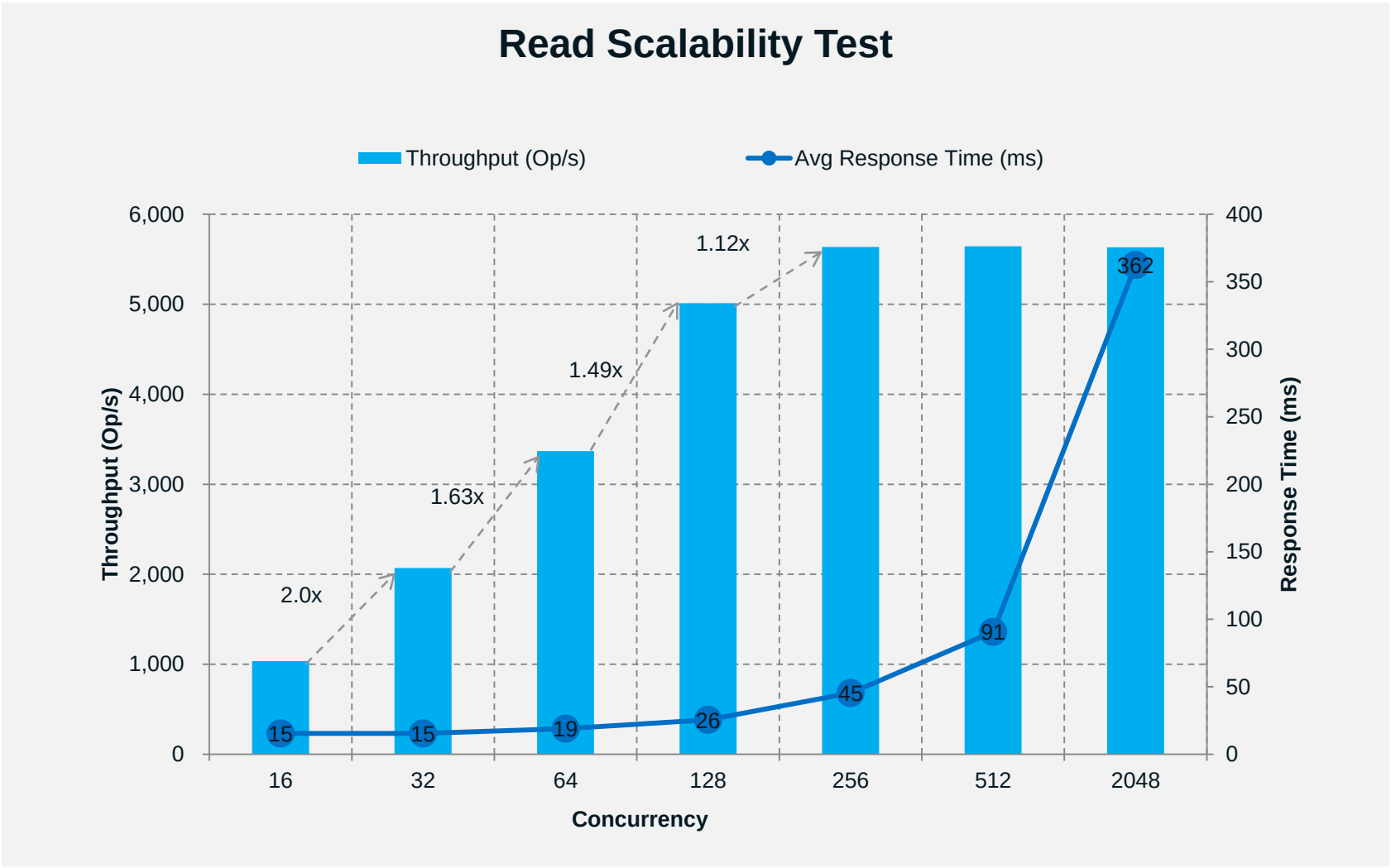
- ratio of successful transmission (%)

Experiments

Hardware	Configuration
Townsend_16	3 chassis, 6 servers
Proxy+Auth Node	
	2x WSM-EX 2.26GHz (HT ON)
	64GB RAM
	1x 250GB SATA disk
Storage Node	
	2x NHM-EP 2.93GHz (HT ON)
	12GB RAM
	12x Seagate 73GB SAS as Storage through one LSI 3801 card.
Networking	1x 10GbE between proxy and storage node, 1x 1GbE per storage node
	10GbE between proxy and Client
Client Node	
	Same as Storage Node except 2x 1GbE bonding used.
Switch	shared with others
Software	
OS	RHEL 6.1 (kernel 2.6.39)
Swift	Diablo (swift 1.4.3)



Read Performance

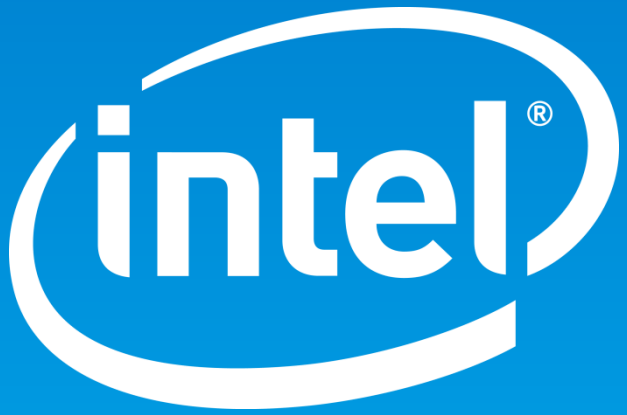


Future work

Workload model

UI improvements

More experiments


intel