Presentation Take-Aways

Know how to:

- Design the Workload
- Use the Object Storage Workload Tools
- Analyze the Results
Agenda

- Motivation and Testing Methodology
- Designing Workloads
- Workload Test Tools (Usage and Results)
- Demo
- Questions
- Backup Slides
Motivation to Develop

Need to incorporate workload-driven requirements into Red Hat Storage development, documentation, test, and release processes. Continuously test and validate those storage workloads going forward.

Object Storage Workload Testing

- Simulate customer production environments
- Scale-out workloads sampled from key customers
- Record client I/O and system resource utilization statistics
Testing Methodology

Comprehensive workload profile:

- Clusters pre-filled to simulate customer production environments
- Ceph RGW scale-out workload sampled from key customers
- Automated failure injection with in-house tooling
- Recording client I/O throughput and latency statistics
- Log Ceph radosgw system resource utilization (cpu, memory)
- Additional logging: fill rates; RGW garbage collection

Workload generated with COSBench

https://github.com/intel-cloud/cosbench
Designing Workloads

Workload Modeling

● Layout
  ○ Number of Buckets
  ○ Number of Objects (per Bucket)
● Object sizes
● Operation types/mixture
● Throughput per day
  ○ Number of objects accessed
  ○ Number of objects modified

Considerations

● Micro benchmarks vs. Production simulation
● Ensure workload generator is not a bottleneck
Designing Workloads

Workload Specifications

● Sizing for Capacity = number of objects
  ○ Test Conditions
    ■ Available capacity - cap (factor in replication type)
    ■ Percent cluster fill - %fill
    ■ object size - objsz
    ■ number of buckets/containers - numbuckets
  ○ numobj = ((cap * %fill) / objsz) / numbuckets

● Sizing for Performance = number of workers
  ○ Target average latency (QoS)
  ○ Perform sizing runs to determine cluster performance level
  ○ Adjust number of workers - numworkers
Micro-benchmarks

Object sizes = Fixed (constant) sizes

- 64K - Small size (Thumbnail images, small files etc.)
- 1M - Medium size (Images, text files etc.)
- 32M - Large size (Data Analytics, HD images, log files, backup etc.)

Test Selection (single operation type)

- Sequential: 100% Sequential Read, then 100% Sequential Write
- Random: 100% Random Read, then 100% Random Write

Typically single operation type and single object size
Production Simulation ‘hybrid’

Object size Histogram

- 1K - 50%
- 64K, 8M, 64M - 15% each
- 1G - 5%

Operation types and mix

- 60% Read
- 10% List
- 16% Delete
- 14% Write

Based on 48 hour production monitoring
Workload Generator Considerations

Scaling factors

- Network performance - Driver to Object Storage
- Number of Drivers
- Workload definition
  - Object sizes
  - Operation type
  - Number of workers/threads (per Driver)

Workload generator limits

- Driver processing overhead (‘mock’ driver info follows)
Workload Generator Limits

COSbench Scaling Measurements:
- Use ‘mock’ storage driver
  ```xml
  <!-- Mock Auth / Storage -->
  <auth type="mock" config="delay=0"/>
  ```

Observations:
- Throughput limiting operation type is Write
- Read, List and Delete deliver higher throughput, but don’t scale with number of workers
- Optimal ratio: 4 Drivers per Client node, each with 3 workers
- On larger configurations use load balancer (HAproxy) between Drivers and Storage
Scaling COSbench

Per Driver Resources (increases with number of workers)
- 1-2x CPU
- 1-2GB Memory RSS

Scaling Number of Drivers
- One Controller with one Driver
  
  ./conf/controller.conf ← one driver section (default)
  
  driver./start-all.sh

- One Controller with X Drivers
  
  Edit ./conf/controller.conf ← lists X driver sections
  
  ./start-driver.sh X
  
  ./start-controller.sh

2019 Storage Developer Conference. © Red Hat. All Rights Reserved.
Scaling COSbench

Example: two Drivers  -  ./conf/controller.conf

[controller]
drivers = 2
log_level = INFO
log_file = log/system.log
archive_dir = archive

[driver1]
name = driver1
url = http://127.0.0.1:18088/driver

[driver2]
name = driver2
url = http://127.0.0.1:18188/driver
Object Storage Testbed Components

Client1
- COSBench Controller + Driver 1

MON

Nodes 1-3

Ceph Public Network

Client2
- COSBench Driver 2

RGW

Node 4

OSD1

..................................................................................................................

Ceph Cluster Network

Client7
- COSBench Driver N

RGW

Node N

OSD1

..................................................................................................................

2019 Storage Developer Conference. © Red Hat. All Rights Reserved.
Object Storage Workload Testing Tools

- genXMLs - generates COSbench workload files
- RGWtest - Ceph RGW testing automation
- RGWOCP - deploys Ceph RGW & COSbench on Kubernetes
- OSDfailure - injects failures and measures Client I/O impact

https://github.com/jharriga/
genXMLs - Overview

Purpose
● Generates workload files and text COSbench results

Automation Capabilities
● Generates workload files (genXMLs.sh)
● Produces COSbench ‘General Report’ (cbparser.py)

Repo URL
● https://github.com/jharriga/genXMLs
genXMLs - Usage

Procedure

- **Install**
  - git clone https://github.com/jharriga/genXMLs

- **Configure**
  - Edit genXMLs.sh
    - akey, skey, endpt ← AUTH
    - testname, runtime
    - objSIZES, numCONT, numOBJ

- **Generate workload files**
  - ./genXMLs.sh

See next slide for workload file inventory
<table>
<thead>
<tr>
<th>FILENAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>fill.xml</td>
<td>cluster fill workload (creates buckets and objects) - first workload</td>
</tr>
<tr>
<td>empty.xml</td>
<td>empty cluster (removes objects and buckets) - last workload</td>
</tr>
<tr>
<td>seqops.xml</td>
<td>performs sequential reads and writes (two workstages)</td>
</tr>
<tr>
<td>randomops.xml</td>
<td>performs random reads and writes (two workstages)</td>
</tr>
<tr>
<td>mixedops.xml</td>
<td>performs mixture of read, list, write and delete operations</td>
</tr>
</tbody>
</table>
RGWtest - Overview

Purpose
● Automates Ceph RGW performance testing

Automation Capabilities
● Generates workload files
● Configures RGW for test runs
  ○ Creates user and pools
  ○ Inserts user credentials in workload files
● Executes and monitors workloads
  ○ logs system resource utilization
  ○ logs Ceph stats

https://github.com/jharriga/RGWtest
RGWtest - Usage

- **Install**
  - git clone https://github.com/jharriga/RGWtest

- **Configure**
  - Edit vars.shinc

- **Run tests**
  - writeXML.sh
  - resetRGW.sh
  - runIOworkload.sh <workload.xml>

- **Review results**
  - logfile (RGWtest/<RESULTSDIR>/<LOGFILE>)
  - client performance (COSbench controller)
## RGWtest - Workload Files

<table>
<thead>
<tr>
<th>FILENAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillWorkload.xml</td>
<td>cluster fill workload (creates buckets and objects) - first workload</td>
</tr>
<tr>
<td>emptyWorkload.xml</td>
<td>empty cluster (removes objects and buckets) - last workload</td>
</tr>
<tr>
<td>ioWorkload.xml</td>
<td>User defined workload built from ‘XMLtemplates’ files</td>
</tr>
</tbody>
</table>

**Execute a workload:**

```
runIOWorkload.sh <workload.xml>
```
RGWtest - Configuration (1 of 2)

Edit vars.shinc - Runtime Environment

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Default Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONhostname</td>
<td>pcloud10</td>
<td>Ceph MON hostname/ip address</td>
</tr>
<tr>
<td>RGWhostname</td>
<td>pcloud08</td>
<td>Ceph radosgw hostname/ip address</td>
</tr>
<tr>
<td>cosPATH</td>
<td>/root/v0.4.2</td>
<td>Path to locally installed COSbench</td>
</tr>
<tr>
<td>rgwUSER</td>
<td>johndoe:swift</td>
<td>Object storage username</td>
</tr>
<tr>
<td>rgwURL</td>
<td>localhost:5000</td>
<td>Object storage endpoint</td>
</tr>
<tr>
<td>preparePTYPE</td>
<td>ec</td>
<td>replication (ec or 3-way)</td>
</tr>
<tr>
<td>pg_data, pg_index, pg</td>
<td>4096, 256, 256</td>
<td>determined by PGCALC</td>
</tr>
</tbody>
</table>
## RGWtest- Configuration (2 of 2)

Edit vars.shinc - Workload definition

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Default Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>objSIZES</td>
<td>histogram</td>
<td>1KB/50%, 8KB/15%, 65MB/15%, 1GB/5%</td>
</tr>
<tr>
<td>numCONT</td>
<td>5</td>
<td>number of containers (or buckets)</td>
</tr>
<tr>
<td>numOBJ</td>
<td>232000</td>
<td>number of objects per container</td>
</tr>
<tr>
<td>numOBJmax</td>
<td>numOBJ</td>
<td>useful for aging runs - ( numOBJ * 2 )</td>
</tr>
<tr>
<td>runtime_sec</td>
<td>3600</td>
<td>1 hour runtime (in seconds)</td>
</tr>
<tr>
<td>fillWORKERS</td>
<td>40</td>
<td>number of workers/threads - fillWorkload.xml</td>
</tr>
<tr>
<td>runtestWORKERS</td>
<td>40</td>
<td>number of workers/threads - ioWorkload.xml</td>
</tr>
</tbody>
</table>
RGWtest - Workload Specifications

Define new workload specifications (advanced)

- RGWtest/Templates contains the template workload files
- Values from vars.shinc are inserted by ‘writeXML.sh’

EXAMPLE - use alternate template

Edit vars.sh → RUNTESTtemplate="TMPL_deleteWrite.tmpl"

writeXML.sh
runIOWorkload ioWorkload.xml

*Users can modify existing or create new templates*
RGWtest
Review Results
RGWtest - logfile

Default location ← RESULTSDIR="./RESULTS"

- LOGFILE="${RESULTSDIR}/${PROGNAME}_${ts}.log"

Collected statistics ($pollinterval="1m")

- ‘ceph df’ capacity - RAW and default.rgw.buckets.data
- System resource utilization
  - OSD/RGW system load average
  - radosgw process and memory stats
  - ceph-osd process and memory stats
- Ceph RGW garbage collection
- Ceph RGW resharding activity

All log entries are timestamped
### RGWtest - Logfile excerpt

#### Logfile excerpt - Fill Cluster 30%

**Start**

<table>
<thead>
<tr>
<th>GLOBAL:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE</td>
<td>AVAIL</td>
</tr>
<tr>
<td>524TiB</td>
<td>523TiB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POOLS:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>ID</td>
</tr>
<tr>
<td>default.rgw.buckets.data</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**End**

<table>
<thead>
<tr>
<th>GLOBAL:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE</td>
<td>AVAIL</td>
</tr>
<tr>
<td>524TiB</td>
<td>379TiB</td>
</tr>
</tbody>
</table>
RGWtest - Review Results

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Conditions</th>
<th>Avg Thruput</th>
<th>Avg Latency</th>
<th>99% Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillCluster 30%</td>
<td>3.1 filestore</td>
<td>147 op/s</td>
<td>466 ms</td>
<td>8640 ms</td>
</tr>
<tr>
<td>fillCluster 30%</td>
<td>3.2 bluestore</td>
<td>164 op/s</td>
<td>416 ms</td>
<td>8260 ms</td>
</tr>
<tr>
<td>PERCENT CHANGE</td>
<td>N/A</td>
<td>+11%</td>
<td>-11%</td>
<td>-4%</td>
</tr>
</tbody>
</table>

Logfile Statistics:
- Load average: 15 min avg - filestore=23.2 ; bluestore=8.7
- CPU usage (PCPU) and memory usage (VSZ and RSS)
  - Ceph-osd
  - Radosgw
- Workload runtime: start and end timestamp
## RGWtest - Review Results

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Conditions</th>
<th>Avg Thruput</th>
<th>Avg Latency</th>
<th>99% Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>hybridSS initial (read stats, 1hr)</td>
<td>3.1 filestore</td>
<td>85 op/s</td>
<td>644 ms</td>
<td>10080 ms</td>
</tr>
<tr>
<td>hybridSS initial (read stats, 1hr)</td>
<td>3.2 bluestore</td>
<td>142 op/s</td>
<td>358 ms</td>
<td>6540 ms</td>
</tr>
</tbody>
</table>

**PERCENT CHANGE**

- N/A
- +67%
- -45%
- -35%

**Logfile Statistics:**
- Load average: 15 min avg - filestore=27.2; bluestore=7.3
- CPU usage (PCPU) and memory usage (VSZ and RSS)
  - Ceph-osd
  - Radosgw
- Workload runtime: start and end timestamp
Tools Demo
Presentation Take Away’s

Know how to:

▪ Design the Workload
▪ Use the Object Storage Workload Tools
▪ Analyze the Results
Questions/Discussion
BACKUP SLIDES
RGWtest - Procedure

- Calculate pg_num values for RGW pools
  - https://access.redhat.com/labsinfo/cephpgc (downstream - supports EC)
  - https://ceph.io/pgcalc/ (upstream)
- Edit RGWtest/vars.shinc
- Create pools and user (resetRGW.sh)
- Write workload files (writeXML.sh)
- Workload run sequence (runIOworkload.sh)
  1. Fill cluster to a predetermined % RAW USAGE
  2. Run hybridSS workload as initial measurement run
  3. Run hybrid2x workload to age the cluster
  4. Run hybridSS workload as aged measurement run
  5. Empty cluster
- Review Results
RGWtest - Test Configuration

Hardware
- 12x OSD nodes (312 OSDs with 500TB RAW)
- 3x MON nodes (one serving as MGR node)
- 17x Client nodes (COSbench drivers)
- 1x Admin node (RGWtest, COSbench controller and ceph-ansible)

Software
- RHEL 7.6
- RHCS 3.2
  - bluestore w/osd_scenario=noncollocated
  - default WAL and DB sizes
- Ceph pool configuration
  - default.rgw.buckets.data: EC 4+2; pg_data=4096
  - All other Ceph pools: 3-way replication; pg_index=256, pg=256
Test Configuration

Ceph Cluster

12x OSD/RGW
6048r’s
- 24x HDD
- 2x NVMe

3x MONs
6018r’s

17x Clients
Dell r620’s
- COSbench Drivers
- HAproxy

1x Admin
Dell r620’s
- RGWtest
- COSbench Controller

12x OSD/RGW
6048r’s
- 24x HDD
- 2x NVMe

OSD1

OSD12

3x MONs
6018r’s

Client1

Client17

Admin

2019 Storage Developer Conference. © Red Hat. All Rights Reserved.
RGWtest - Variable Settings

Object count calculations (sizing capacity)

- Actual available (factoring replication type)
  - 'ceph df' output, the MAX AVAIL for `default.rgw.buckets.data` 332TiB

- RGW Object size distribution
  - `objSIZES="h(1|1|50,64|64|15,8192|8192|15,65536|65536|15,1048576|1048576|5)KB"`

- 62MB Mean Objsz: 
  - `(1*.5)+(64*.15)+(8192*.15)+(65536*.15)+(1048576*.05)`

- `numobj = ((cap * %fill) / objsz) / numbuckets`
  - `((332TiB * 0.3) / 62MB) / 5 = ~400k`

RGWtest variable settings (vars.shinc)

- `preparePTYPE=ec, k=4, m=2` ← EC 4+2 replication
- `pg_data=4096, pg_index=256, pg=256` ← [https://access.redhat.com/labs/cephpgc/](https://access.redhat.com/labs/cephpgc/)
- `numCONT=5, numOBJ=400000` ← 2M Objects total
- `fillWORKERS=68` ← 17 driver nodes (four per driver node)
- `runtestWORKERS=68` ← 17 driver nodes (four per driver node)
## Ceph Placement Groups per Pool Calc.

### Step 2. Select a Ceph use case

- Rados Gateway Only -- Use for S3 and/or Swift workloads only

### Step 3. Select special conditions (optional)

- **Support Erasure Coding (EC)**
  
  Supported only for the RGW and native librados object storage.

### Step 4. Adjust values for pools and PGs

Set values for all pools

<table>
<thead>
<tr>
<th>Pool Name</th>
<th>Pool Type</th>
<th>Size</th>
<th>OSD #</th>
<th>%Data</th>
<th>Target PGs per OSD</th>
<th>Suggested PG Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>.rgw.root</td>
<td>Replicated</td>
<td>3</td>
<td>312</td>
<td>0.10</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>default:rgw.intent-log</td>
<td>Replicated</td>
<td>3</td>
<td>312</td>
<td>0.10</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>default:rgw.log</td>
<td>Replicated</td>
<td>3</td>
<td>312</td>
<td>0.10</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>default:rgw.buckets.data</td>
<td>Erasure Coding</td>
<td>K=4</td>
<td>M=2</td>
<td>94.80</td>
<td>100</td>
<td>4096</td>
</tr>
<tr>
<td>default:rgw.buckets.extra</td>
<td>Replicated</td>
<td>3</td>
<td>312</td>
<td>1.00</td>
<td>100</td>
<td>256</td>
</tr>
<tr>
<td>default:rgw.buckets.index</td>
<td>Replicated</td>
<td>3</td>
<td>312</td>
<td>3.00</td>
<td>100</td>
<td>256</td>
</tr>
</tbody>
</table>
# Workload

## Basic Info

**ID:** w13  **Name:** hybrid2x  **Current State:** finished

**Submitted At:** Jun 9, 2019 9:19:03 PM  **Started At:** Jun 9, 2019 9:19:03 PM  **Stopped At:** Jun 11, 2019 9:19:15 PM

[more info](#)

## Final Result

### General Report

<table>
<thead>
<tr>
<th>Op-Type</th>
<th>Op-Count</th>
<th>Byte-Count</th>
<th>Avg-ResTime</th>
<th>Avg-ProcTime</th>
<th>Throughput</th>
<th>Bandwidth</th>
<th>Succ-Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>4.46 mops</td>
<td>237.03 TB</td>
<td>616.61 ms</td>
<td>50.02 ms</td>
<td>25.79 op/s</td>
<td>1.37 GB/S</td>
<td>99.49%</td>
</tr>
<tr>
<td>list</td>
<td>740.94 kops</td>
<td>0 B</td>
<td>5.95 ms</td>
<td>5.95 ms</td>
<td>4.29 op/s</td>
<td>0 B/S</td>
<td>99.48%</td>
</tr>
<tr>
<td>write</td>
<td>1.19 mops</td>
<td>63.26 TB</td>
<td>574.48 ms</td>
<td>78.97 ms</td>
<td>6.91 op/s</td>
<td>366.11 MB/S</td>
<td>100%</td>
</tr>
<tr>
<td>delete</td>
<td>1.04 mops</td>
<td>0 B</td>
<td>16.71 ms</td>
<td>16.71 ms</td>
<td>6.05 op/s</td>
<td>0 B/S</td>
<td>100%</td>
</tr>
</tbody>
</table>

### ResTime (RT) Details

<table>
<thead>
<tr>
<th>Op-Type</th>
<th>60%-RT</th>
<th>80%-RT</th>
<th>90%-RT</th>
<th>95%-RT</th>
<th>99%-RT</th>
<th>100%-RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>&lt; 80 ms</td>
<td>&lt; 250 ms</td>
<td>&lt; 820 ms</td>
<td>&lt; 1,020 ms</td>
<td>&lt; 12,700 ms</td>
<td>&lt; 16,870 ms</td>
</tr>
<tr>
<td>list</td>
<td>&lt; 10 ms</td>
<td>&lt; 10 ms</td>
<td>&lt; 10 ms</td>
<td>&lt; 20 ms</td>
<td>&lt; 50 ms</td>
<td>&lt; 1,530 ms</td>
</tr>
<tr>
<td>write</td>
<td>&lt; 70 ms</td>
<td>&lt; 250 ms</td>
<td>&lt; 790 ms</td>
<td>&lt; 980 ms</td>
<td>&lt; 11,870 ms</td>
<td>&lt; 17,500 ms</td>
</tr>
<tr>
<td>delete</td>
<td>&lt; 10 ms</td>
<td>&lt; 30 ms</td>
<td>&lt; 50 ms</td>
<td>&lt; 70 ms</td>
<td>&lt; 120 ms</td>
<td>&lt; 10,110 ms</td>
</tr>
</tbody>
</table>

---

2019 Storage Developer Conference. © Red Hat. All Rights Reserved.
**ID:** w15  **Name:** delWrite2hr  **Current State:** finished

**Submitted At:** Jun 11, 2019 10:20:35 PM  **Started At:** Jun 11, 2019 10:20:35 PM  **Stopped At:** Jun 12, 2019 12:20:48 AM

more info

### Final Result

### General Report

<table>
<thead>
<tr>
<th>Op-Type</th>
<th>Op-Count</th>
<th>Byte-Count</th>
<th>Avg-ResTime</th>
<th>Avg-ProcTime</th>
<th>Throughput</th>
<th>Bandwidth</th>
<th>Succ-Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>305.56 kops</td>
<td>16.13 TB</td>
<td>462.57 ms</td>
<td>62.69 ms</td>
<td>42.45 op/s</td>
<td>2.24 GB/S</td>
<td>100%</td>
</tr>
<tr>
<td>delete</td>
<td>203.46 kops</td>
<td>0 B</td>
<td>12.7 ms</td>
<td>12.7 ms</td>
<td>28.27 op/s</td>
<td>0 B/S</td>
<td>100%</td>
</tr>
</tbody>
</table>

### ResTime (RT) Details

<table>
<thead>
<tr>
<th>Op-Type</th>
<th>60%-RT</th>
<th>80%-RT</th>
<th>90%-RT</th>
<th>95%-RT</th>
<th>99%-RT</th>
<th>100%-RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>&lt; 20 ms</td>
<td>&lt; 240 ms</td>
<td>&lt; 660 ms</td>
<td>&lt; 730 ms</td>
<td>&lt; 8,770 ms</td>
<td>&lt; 12,450 ms</td>
</tr>
<tr>
<td>delete</td>
<td>&lt; 20 ms</td>
<td>&lt; 20 ms</td>
<td>&lt; 20 ms</td>
<td>&lt; 70 ms</td>
<td>&lt; 10,040 ms</td>
<td></td>
</tr>
</tbody>
</table>
END