COSBench User Guide

Version 2.8.5

Nov., 2014

Wang, Yaguang

This document describes how to install, configure, and run COSBench (a cloud storage benchmark tool) step by step, explains how to define workloads using configuration files, and provides reference examples.

Document Number: 328791-001US

0413/RJM/MESH/PDF 328791-001US

Contents

Revision History

Revision	Date	Description
0.5	July 14, 2012	Initial version
0.6	July 18, 2012	Add “init” and “dispose” stages in AmpliStor* example and description for special stages
0.7	July 20, 2012	Add “nsroot” to storage parameter list to access AmpliStor v2.5 namespace; by default, it’s “/namespace”, set to “/manage/namespace” for v2.5
0.8	July 24, 2012	Change default listening ports: • 8088->18088 • 8089->18089 • 9088->19088 • 9089->19089
0.9	August 1, 2012	Change example port numbers to 19088 and 18088 to avoid confusion
1.0	August 9, 2012	Add one section to describe data results and one section for FAQs
1.1	August 13, 2012	• Add one paragraph in result to explain metrics • Modify AmpliStor sample to reflect v2.5 needs • Add parameter list
1.2	August 24, 2012	Enhance content based on internal and external user feedback
1.3	August 30, 2012	• Add Red Hat screenshot • Change runtime from 60 to 300 for AmpliStor example to avoid confusion • Remove internal link for package downloading • Fix one bug in “cleanup” stage of Swift sample
1.4	September 14, 2012	Fix inconsistencies
1.5	September 17, 2012	Change default OS to Ubuntu* 12.04.1 LTS desktop
1.6	November 2, 2012	Major modifications: • Transfer all scripts to Ubuntu 12.04.1 compatible • Add OS installation steps • Add object integrity check parameter • Add details about selector description • Add details about directory structure • Move workload configuration section from Appendix A to main body

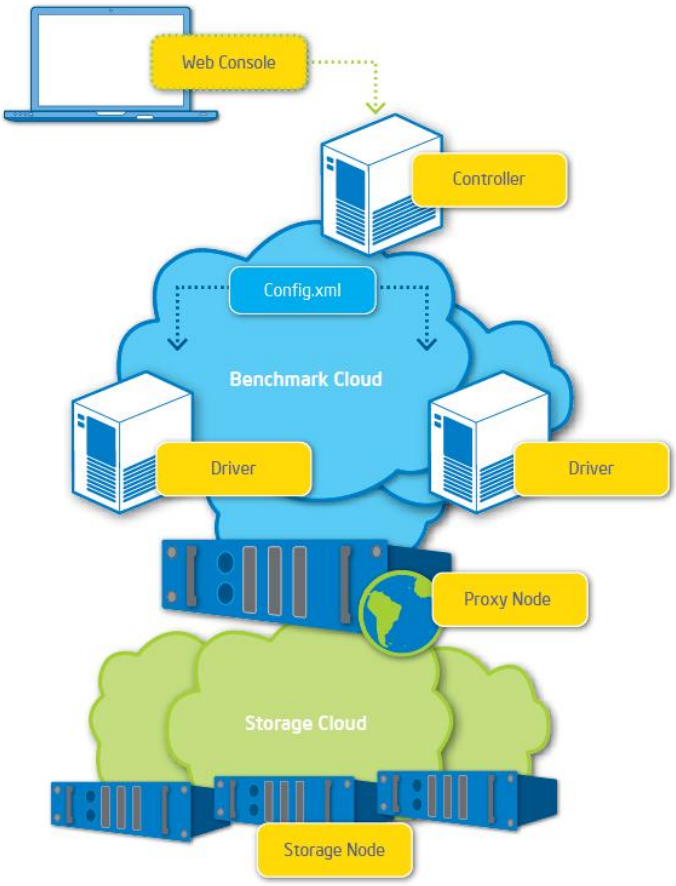
Revision	Date	Description
1.7	November 13, 2012	Minor modifications: • Correct batch script names • Add one item in FAQ for handling “OOM” error
1.8	November 20, 2012	Change parameter “url” to “auth_url” for swauth and keystone to avoid confusion
1.9	January 14, 2013	• Add parameter “tenant_name” for keystone • Add items in FAQ to explain testing with large objects
2.0	January 25, 2013	• Correct two minor typographic errors • Add explanation about histogram data • Reword FAQ #12
2.1	February 19, 2013	• Constrain supported AmpliStor versions to v2.3 and v2.5 • Minor formatting modifications
2.2	March 7, 2013	• Fix one typo from “apt-get” to “apt-get install” • Correct one word from “turn-around point” to “tipping point” • Add section 6.1.14 for how to split read/write • Enhance section 6.1.6 for how to reuse data • Minor rewording from “policy” to “policy UID” in section 6.3.1 • Enhance section 6.1.7 for configuring multiple same stages • Add section 6.3.3 for how to simplify policy UID setting
2.3	March 8, 2013	• Add explanation about using “ps” in section 3.7 • Add explanation about how to do pre-test in section 6.2.2 • Reword section 6.1.12 to explain conditions for using only one worker • Replace “Excel” with “spreadsheet program” in section 5.2.2 • Add case for multiple client daemons in AmpliStor section of Appendix A • Add explanation for what commands do in section 2.2.2 • Add example controller configuration to show multiple drivers supporting in section 3.2.1 • Correct one typographic error in section 6.1.9
2.4	March 13, 2013	• Change “enlarging” to “expanding” and add example

Revision	Date	Description
2.5	March 29, 2013	• Change “127.0.0.1” to “192.168.250.36” in text and screen captures of sections 2.4.3, 3.5, 3.6, 4.4.3, and Appendix A • Add AmpliStor v3.1 in section 1.1 • Remove mention of licensing in section 2.3.1
2.6	April 7, 2013	• Add package list information in section 2.1 and provide one separate pkg.lst • Add “retry” parameter for auth to overcome failures at high concurrent requests for authentication. • Change “mandatory” to “required” in section 3.2.1
2.7	July 22, 2013	• Add filewrite usage information in section 4.9 (from Niklas) • Add sequential selector in section 4.2.2 (from Niklas) • Change the measurement unit for bandwidth from MiB/s (=1024*1024 Byte/s) to MB/s (=1000*1000 Byte/s).
2.7.1	July 29, 2013	• Add S3 configuration information in section 4.5.3 • Add S3 FAQ in section 6.4 • Change mailing list url in section 1 • Add supported storage system matrix in section 1.3
2.7.2	August 27, 2013	• Minor rewording
2.8	October 31, 2013	• Add direct auth for swift in section 4.5.3 • Add sproxyd adapter parameter list in section 4.5.3 • Add “delay” stage in section 4.8.2 • Add tomcat related configuration information, including web authentication for web console in new section 3.8 • Add configurable “archive” folder in section 3.2 • Add workload management information in section 3.9 • Add sentences about multiple stages in workload configuration UI page in section 3.6.1 • Add histogram selector in section 4.2.2
2.8.1	November 14, 2013	• Add section 4.10 to describe additional hints like data division strategy for workload configuration. • Correct sentences in section 4.2.2 for range selector, • Add FAQ 6.1.17 to explain how to use range selector in normal stage. • Add “afr” parameter in section 4.7.2

Revision	Date	Description
2.8.2	Feb 13, 2014	• Add parameter “pool_size” for Scality sproxyd adapter in section 4.5.3. • Add explanation for “nsroot” parameter in section 6.3.4. • Update the binary package link in section 2.2.1 • Correct description for histogram selector to emphasis the number is weight instead of percentage in section 4.2.2. • Add “httpauth” authentication mechanism in section 4.4. • Add “cdmi” storage type in section 4.5.
2.8.3	May 12, 2014	• Added advanced config UI related information in section 3.6.1 and 3.6.2 • Add cvstool as optional prerequisites if user may need it to process generated results. • Add “driver” parameter in section 4.7.2. • Add section 6.5 for ceph FAQ. • Update compatibility list in section 1.3. • Add section 5.2.4 for response time breakdown information. • Add section 6.6 for CDMI FAQ. • Add cdmi_swift in section 4.5.3. • Add FAQ in 6.2.4 for tempauth.
2.8.4	July 3, 2014	• Add “policy” parameter in swift for storage policy feature in section 4.5.3. • Add one FAQ entry to explain how to enable storage policy for swift in section 6.2.5 • Correct the extension of tomcat configuration files from .conf to .xml in section 3.8 • Add one FAQ entry to explain how to make multiple drivers running on the same physical node in section 6.1.18.
2.8.5	Oct. 22, 2014	• Add librados storage parameters in section 4.5.3.1.9 • Add FAQ entry to explain how to change logging level to disclose more detailed information for trouble shooting in section 6.1.19 • Add “caching” parameter for Auth in section 4.4.3 • Add new “list” operator in section 4.9.3

1 Introduction

COSBench is a distributed benchmark tool to test cloud object storage systems, it supports a few cloud object storage systems so far (see 1.3 “Supported Cloud Object Storage System Matrix”). COSBench also allows users to create adaptors for additional storage systems. Please refer to the “COSBench Adaptor Development Guide” for details.



COSbench consists of two key components:

- Driver (also referred to as COSBench Driver or Load Generator):
    - Responsible for workload generation, issuing operations to target cloud object storage, and collecting performance statistics.
    - Can be accessed via http://<driver-host>:18088/driver/index.html.
- Controller (also referred to as COSBench Controller):
    - Responsible for coordinating drivers to collectively execute a workload, collecting and aggregating runtime status or benchmark results from driver instances, and accepting workload submissions.
    - Can be accessed via http://<controller-host>:19088/controller/index.html.

The controller and driver can be deployed on the same node or different nodes, and the node can be a physical machine or virtual machine (VM) instance.

Intel source code for COSBench is being released under the Apache 2.0 license, and hosted at http://github.com/intel-cloud/cosbench/.

A mailing list has been established for COSBench at the following location: http://cosbench.1094679.n5.nabble.com/.

1.1 Reference Hardware Configuration

The hardware configurations used for validation purposes in Intel labs are given below. This information is provided for reference only, as the appropriate systems for various implementations are highly dependent upon individual usage scenarios. Also note that network resources play a vital role in COSBench implementations.

Hardware	Configuration
Controller	
Processor	Two Intel® Xeon® processors X5570 @ 2.93 GHz
RAM	12 GB RAM
Storage	1x 120 GB+ disk drive
Network	Intel® 82574 Gigabit Ethernet Controller
Driver	
Processor	Two Intel Xeon processors X5570 @ 2.93 GHz
RAM	12 GB RAM
Storage	1x 50 GB+ disk drive
Network	Intel® 82599 10 Gigabit Ethernet Controller

1.2 System Requirements

NOTE: The current release of COSBench features Ubuntu* 12.04.1 LTS Desktop, but the COSBench development team assumes that organizations will install using various OSs and contribute related feedback to the community.

- Ubuntu 12.04.1 LTS Desktop
- Java* Runtime Environment 1.6 or later
- Curl 7.22.0 or later
- Csvtool if processing generated csv files is required.
- Free TCP port (ensure these ports are accessible non-locally):

- On COSBench controller machine: 19088
- On COSBench driver machines: 18088

NOTE: Throughout this document, command line is **bolded** and *italicized;* yellow text is used for emphasis, to draw attention to specific information.

1.3 Supported Cloud Object Storage System Matrix

Generally, two parts will be involved to access each cloud object storage system, they are the authentication mechanism and object access semantics. To meet the complexity from different systems, COSBench treats them separately, and encapsulate into two APIs (AuthAPI and StorageAPI).

Developer can implement them in different bundles, or combine them into one, and users can combine one Auth API implementation with multiple storage API implementations, or associate multiple Auth API implementations with one storage API implementations. Please refer to the "COSBench Adaptor Development Guide" for details.

Below table lists the status of different AuthAPI and StorageAPI combinations so far, it may be updated time to time:

Cloud Storage	Auth	Status
OpenStack* Swift	tempauth	OK
	swauth	OK
	keystone	OK
	direct	OK
Amplidata* Amplistor	none	OK
	basic/digest	Verifying
Ceph	librados	OK
	rados GW (swift)	OK
	rados GW (s3)	OK
Amazon* S3	integrated	OK
Scality	sproxyd	OK
CDMI		
CDMI-base	basic/digest	OK
CDMI-swift	swauth/keystone	OK
None	none	OK
Mock	mock	OK

Note:

* librados is contributed by Niklas Goerke - niklas974@github,
* sproxyd is contributed by Christophe Vedel from Scality

1.4 What's in the Rest of This Guide

This document describes how to install, configure, and use COSBench, a cloud storage benchmarking tool.

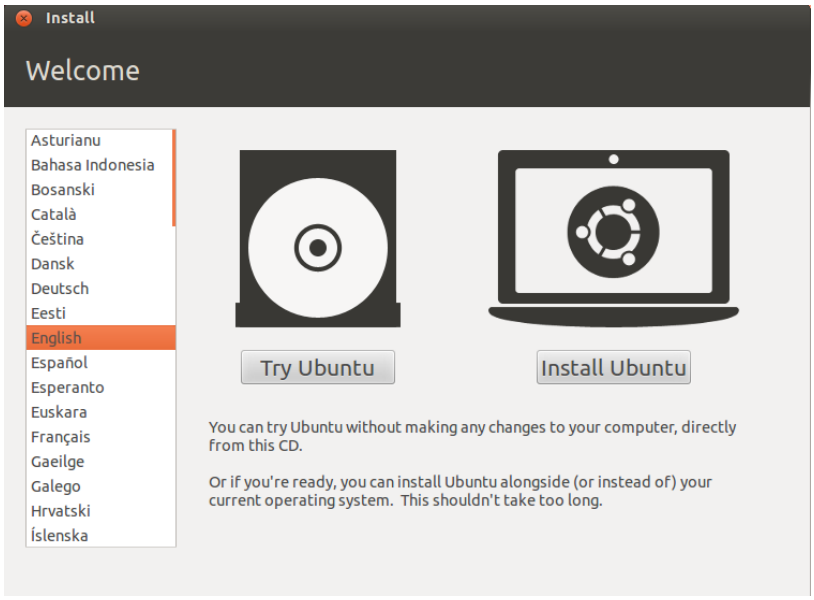
- Section 2 covers the initial installation and testing of COSBench.
- Section 3 explains how to configure and run the tool.
- Section 4 instructs the user about how to define workloads.
- Section 5 explains the results provided by COSBench and how to interpret them.
- Section 6 answers frequently asked questions.

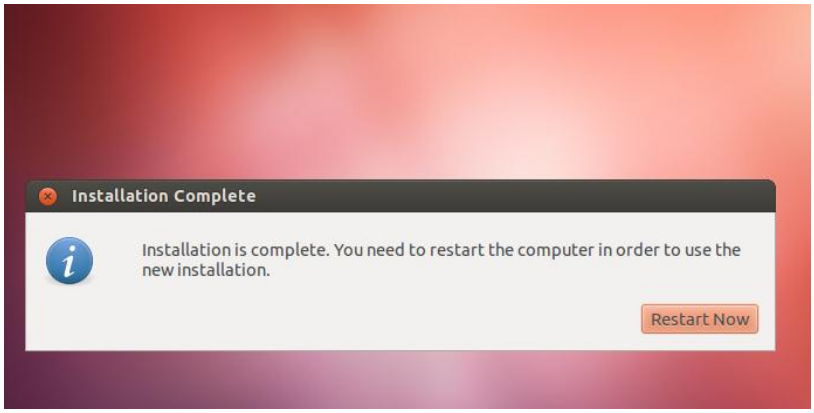
- Appendix A provides sample configurations for different storage systems.

2 Installing COSBench

2.1 Installing the Operating System

1. Download Ubuntu Desktop 12.04.1 LTS.
2. Follow the instructions in the Ubuntu installation guide.
3. Below are screenshots from major steps during installation, which include the creation of one user named "cosbench"; all other settings may be left at their defaults or modified at the user's discretion.





4. The final package list after installation can be found in the file "pkg.lst" on the github site.

2.1.1 Installing the Java Runtime Environment (JRE)

- OpenJDK is the default JRE; Oracle JRE should also work.
- If an Internet connection is available, the package can be installed through apt-get as follows:

```
cosbench@cosbox:~$ sudo apt-get update
cosbench@cosbox:~$ sudo apt-get install openjdk-7-jre
```

- If no Internet connection is available, the JRE can be installed using Debian* software packages; two packages are essential: JRE-LIB and JRE-HEADLESS.
- Those packages can be installed as follows (this procedure uses "/tmp" as an example; a different folder may be used at the user's discretion):

```
cosbench@cosbox:/tmp$ sudo dpkg –i –force depends openjdk-7-jre-lib_7u7-2.3.2a-0ubuntu0.12.04.1_all.deb
(Reading database …

cosbench@cosbox:/tmp$ sudo dpkg –i –force depends openjdk-7-jre-headless_7u7-2.3.2a-0ubuntu0.12.04.1_amd64.deb
Selecting previously unselected package openjdk-7-jre-headless …

cosbench@cosbox:/tmp$ java –showversion
java version "1.7.0_07"
…
```

2.1.2 Installing Curl

- If an Internet connection is available, Curl can be installed as follows:

```
cosbench@cosbox:~$ sudo apt-get update
cosbench@cosbox:~$ sudo apt-get install curl
```

- If no Internet connection is available, install Curl using Debian software packages:

```
cosbench@cosbox:/tmp$ sudo dpkg –i curl_7.22.0-3ubuntu4_amd64.deb

cosbench@cosbox:/tmp$ curl –V
curl 7.22.0 (x86_64-pc-linux-gnu) …
```

2.2 Installing COSBench

2.2.1 Preparation

In the current release, the COSBench controller and driver are combined; they do not each have a separate package.

Obtain the installation package **<version>.zip** (e.g., 2.1.0.GA.zip) from https://github.com/intel-cloud/cosbench/releases and place it at COSBench package under the home directory on the controller node.

2.2.2 Installation

Follow the commands below to finish the installation, which unpacks the COSBench package into one folder, create one symbolic link called “cos” to it, and make all bash scripts executable:

```
cosbench@cosbox:/tmp$ cd ~
cosbench@cosbox:~$ unzip 2.1.0.GA.zip
cosbench@cosbox:~$ rm cos
cosbench@cosbox:~$ ln –s 2.1.0.GA/ cos
cosbench@cosbox:~$ cd cos
cosbench@cosbox:~$ chmod +x *.sh
```

2.3 Directory Structure

```
drwxrwxr-x  8 cosbench cosbench   4096 Oct 30 23:49 ./
drwxr-xr-x 22 cosbench cosbench   4096 Oct 30 23:21 ../
-rw-rw-r--  1 cosbench cosbench 178499 Oct 24 11:51 3rd-party-licenses.rtf
drwxrwxr-x  2 cosbench cosbench   4096 Oct 30 23:49 archive/
-rw-rw-r--  1 cosbench cosbench   1447 Oct 25 15:56 CHANGE-LOG
-rwxrwxr-x  1 cosbench cosbench   1246 Oct 24 11:51 cli.sh*
drwxrwxr-x  2 cosbench cosbench   4096 Oct 24 11:51 conf/
-rwxrwxr-x  1 cosbench cosbench   1964 Oct 24 11:51 cosbench-start.sh*
-rwxrwxr-x  1 cosbench cosbench    643 Oct 24 11:51 cosbench-stop.sh*
-rw-rw-r--  1 cosbench cosbench 941360 Oct 25 14:39 COSBenchUserGuide.pdf
-rw-rw-r--  1 cosbench cosbench  32256 Oct 24 11:51 Intel\ License.doc
drwxrwxr-x  2 cosbench cosbench   4096 Oct 30 23:49 log/
drwxrwxr-x  2 cosbench cosbench   4096 Oct 30 23:49 main/
-rw-rw-r--  1 cosbench cosbench    112 Oct 24 11:51 notice.txt
drwxrwxr-x  5 cosbench cosbench   4096 Oct 24 11:51 osgi/
-rw-rw-r--  1 cosbench cosbench   1321 Oct 24 11:51 README
-rw-rw-r--  1 cosbench cosbench     89 Oct 24 11:52 start-all.bat
-rwxrwxr-x  1 cosbench cosbench    114 Oct 24 11:51 start-all.sh*
-rw-rw-r--  1 cosbench cosbench    170 Oct 24 11:52 start-controller.bat
-rwxrwxr-x  1 cosbench cosbench    382 Oct 24 11:51 start-controller.sh*
-rw-rw-r--  1 cosbench cosbench    162 Oct 24 11:52 start-driver.bat
-rwxrwxr-x  1 cosbench cosbench    461 Oct 24 11:51 start-driver.sh*
-rwxrwxr-x  1 cosbench cosbench    111 Oct 24 11:51 stop-all.sh*
-rwxrwxr-x  1 cosbench cosbench    200 Oct 24 11:51 stop-controller.sh*
-rwxrwxr-x  1 cosbench cosbench    192 Oct 24 11:51 stop-driver.sh*
-rw-rw-r--  1 cosbench cosbench     50 Oct 24 11:52 web.bat
drwxrwxr-x  3 cosbench cosbench   4096 Oct 30 23:49 work/
```

2.3.1 Scripts

Script	Description
start-all.sh stop-all.sh	Start/stop both controller and driver on current node
start-controller.sh stop-controller.sh	Start/stop controller only on current node
start-driver.sh stop-driver.sh	Start/stop driver only on current node
cosbench-start.sh cosbench-stop.sh	Internal scripts called by above scripts
cli.sh	Manipulate workload through command line

A few Windows* batch scripts are also included, for demonstration purposes only.

Script	Description
start-all.bat	Start both controller and driver on current node
start-controller.bat	Start controller only on current node
start-driver.bat	Start driver only on current node
Web.bat	Open controller web console through locally installed browser

2.3.2 Sub-directories

Sub-directory	Description
archive	Stores all generated results; see the Results section of this document
conf	Configuration files, including COSBench configurations and workload configurations
log	Runtime log files; the important one is system.log
osgi	Contains COSBench libraries and third-party libraries
main	Contains the OSGi launcher

2.4 Verifying Install

The following steps launch the controller and driver on the current node and test to ensure that the installation is correct.

2.4.1 Launching COSBench

HTTP proxy breaks the interaction between controller and driver. To avoid HTTP requests routing, you need to **bypass** the proxy setting:

```
cosbench@cosbox:~$ unset http_proxy
```

Start up the COSBench driver and controller on the current node. By default, the COSBench driver listens on port 18088, and the COSBench controller listens on port 19088.

```
cosbench@cosbox:~$ sh start-all.sh
```

```
cosbench@cosbox:~/cos$ sh start-all.sh
Launching osgi framwork ...
Successfully launched osgi framework!
Booting cosbench driver ...
..
Starting    cosbench-log_2.0    [OK]
..
Starting    cosbench-tomcat_2.0    [OK]
Starting    cosbench-config_2.0    [OK]
Starting    cosbench-core_2.0    [OK]
Starting    cosbench-core-web_2.0    [OK]
Starting    cosbench-api_2.0    [OK]
Starting    cosbench-mock_2.0    [OK]
Starting    cosbench-ampli_2.0    [OK]
Starting    cosbench-swift_2.0    [OK]
Starting    cosbench-keystone_2.0    [OK]
Starting    cosbench-driver_2.0    [OK]
Starting    cosbench-driver-web_2.0    [OK]
Successfully started cosbench driver!
Listening on port 0.0.0.0/0.0.0.0:18089 ...
Persistence bundle starting...
Persistence bundle started.
-----------------------------------------
!!! Service will listen on web port: 18088 !!!
-----------------------------------------
```

```
Launching osgi framwork ...
Successfully launched osgi framework!
Booting cosbench controller ...
...
Starting    cosbench-log_2.0    [OK]
.
Starting    cosbench-tomcat_2.0    [OK]
Starting    cosbench-config_2.0    [OK]
Starting    cosbench-core_2.0    [OK]
Starting    cosbench-core-web_2.0    [OK]
Starting    cosbench-controller_2.0    [OK]
Starting    cosbench-controller-web_2.0    [OK]
Successfully started cosbench controller!
Listening on port 0.0.0.0/0.0.0.0:19089 ...
Persistence bundle starting...
Persistence bundle started.
-----------------------------------------
!!! Service will listen on web port: 19088 !!!
-----------------------------------------
```

2.4.2 Checking Controllers and Drivers

cosbench@cosbox:~$ ***netstat –an \|grep LISTEN \|grep 19088*** # check controller.

```
tcp        0      0 :::19088                :::*        LISTEN
```

Cosbench@cosbox:~$ ***netstat –an \|grep LISTEN \|grep 18088*** # check driver

```
tcp        0      0 :::18088                :::*        LISTEN
```

2.4.3 Testing the Install

```
Cosbench@cosbox:~$ sh cli.sh submit conf/workload-config.xml # run mock test.
Accepted with ID: w1

cosbench@cosbox:~$ sh cli.sh info
Drivers:
driver1 http://127.0.0.1:18088/driver
Total: 1 drivers

Active Workloads:
W1      Thu Jul 12 04:37:31 MST 2012    PROCESSING
```

Open http://127.0.0.1:19088/controller/index.html in a browser to monitor status. In the example below, one "processing" workload is listed in the "active workloads" section.

COSBench is now successfully installed on the current node. Optionally, the workload may be cancelled and COSBench may be stopped as follows:

```
cosbench@cosbox:~$ sh cli.sh cancel w1
W1      Thu Jul 12 23:34:14 MST 2012    CANCELLED

cosbench@cosbox:~$ sh stop-all.sh
Stopping cosbench controller ...
Successfully stopped cosbench controller.

==================================================
Stopping cosbench driver ...
Successfully stopped cosbench driver.
```

2.5 Deploying COSBench

- Copy <version>.zip to the remaining COSBench nodes by means such as scp or shared folder.
- Repeat the procedures listed above for installing COSBench and verifying the installation on each node.

3 Configuring and Running

3.1 General

The COSBench controller and driver depend on different system configuration files to start up, and those configuration files are only for COSBench itself, as opposed to workload configuration.

The following table gives an overview of all the configurations COSBench expects.

Configuration	Description	File Path
controller	Configuration for a controller; read by the controller during its initialization	conf/controller.conf
driver	Configuration for a driver; read by the driver during its initialization	conf/driver.conf
workload	Configuration for a workload being submitted	Submitted via controller's web interface

3.2 Configuring the Controller

3.2.1 Conf/controller.conf

An INI format file is *required* for configuration of the COSBench controller, as in the following example:

```
[controller]
concurrency=1
drivers=3
log_level = INFO
log_file = log/system.log
archive_dir = archive

[driver1]
name=driver1
url=http://192.168.10.1:18088/driver

[driver2]
name=driver2
url=http://192.168.10.2:18088/driver

[driver3]
name=driver3
```

```
url=http://192.168.10.3:18088/driver
```

[controller]

Parameter	Type	Default	Comment
drivers	Integer	1	Number of drivers controlled by this controller
concurrency	Integer	1	Number of workloads that can be executed simultaneously
log_level	String	“INFO”	“TRACE”, “DEBUG”, “INFO”, “WARN”, “ERROR”
log_file	String	“log/system.log”	Where the log file is stored
archive_dir	String	“archive”	Where the archived workload results are stored

The driver section for the nth driver should be named **driver<n>** in order to be recognized.

[driver<n>]

Parameter	Type	Default	Comment
name	String		Label used to identify the driver node. Note that driver name is not necessarily the node’s hostname
url	String		Address to access the driver node

3.3 Configuring the Driver

3.3.1 Conf/driver.conf

This file is optional; the COSBench driver can start up without this configuration file, although the web console can’t correctly label the driver node. Configuration is an INI format file, as in the following example:

```
 [driver]
name=driver1
url=http://192.168.0.11:18088/driver
```

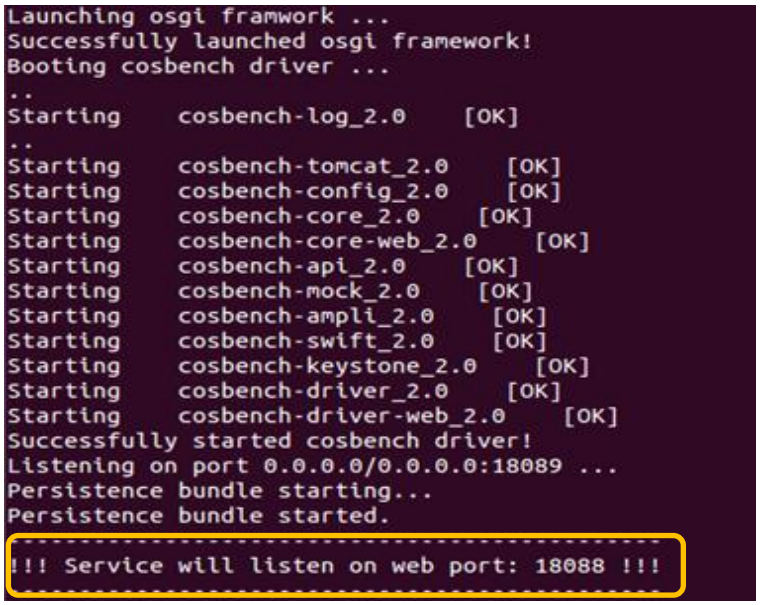
[driver]

Parameter	Type	Default	Comment
name	String		Label used to identify the driver node; note that driver name is not necessarily the node's hostname
url	String		Address to access the driver node

3.4 Starting Drivers

- Edit conf/driver.conf on driver nodes, if desired.
- By default, COSBench driver listens on port 18088.
- Launch driver on all driver nodes.

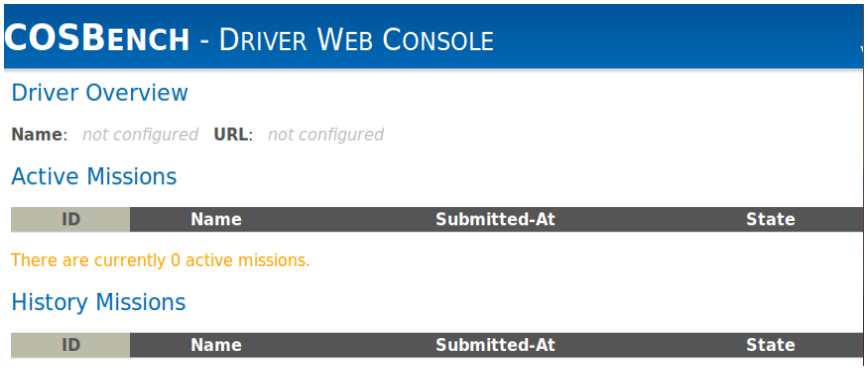
```
sh start-driver.sh
```



- Ensure that all drivers are accessible from the controller using an HTTP connection.
  - By connecting with Curl, one valid HTML file is expected in the console:

```
curl http://<driver-host>:18088/driver/index.html
```

  - When http://<driver-host>:18088/driver/index.html is opened in a web browser, the following web page displays:

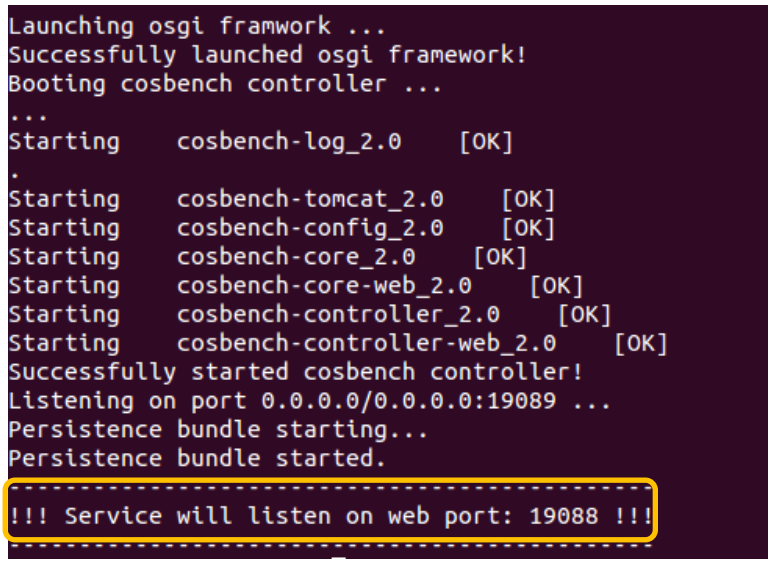


NOTE: If any errors or unexpected results occur, please check system configurations; common issues include firewall filtering or http proxy routing.

3.5 Starting Controllers

- Edit conf/controller.conf on the COSBench controller machine.
- By default, the COSBench controller listens on port 19088**.**
- Launch Controller on the controller node.

```
sh start-controller.sh
```



- Ensure that the controller is started successfully.
  - By connecting with Curl, one valid HTML file is expected in the console:

```
curl http://<controller-host>:19088/controller/index.html
```

- When http://<controller-host>:19088/controller/index.html is opened in a web browser, the following web page displays (note that the <controller-host> IP address 192.168.250.36 shown in the screen capture below is replaced with the actual IP address of the controller node):

3.6 Submitting Workloads

A few templates are provided for reference in the conf/ directory:

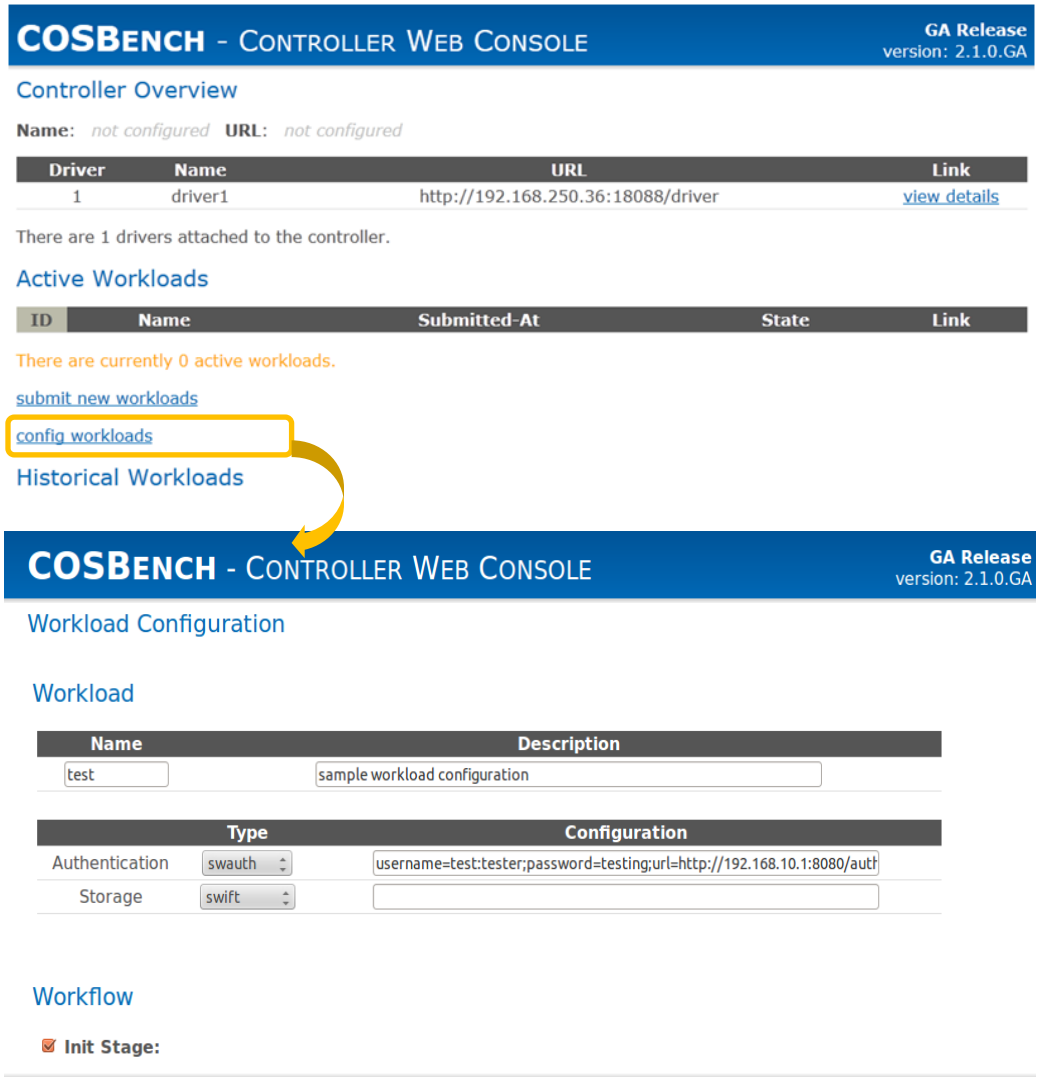
- **workload-config.xml** is a template with comments to describe how to configure for different storage types. It will access mock storage to help with verification.
- **swift-config-sample.xml** is a template for the OpenStack Swift storage system.
- **ampli-config-sample.xml** is a template for the Amplidata AmpliStor v2.3 and v2.5 storage systems. See Appendix A for version-specific configuration information.
- **s3-config-sample.xml** is a template for Amazon S3 compatible storage system.

3.6.1 Defining Workloads

For details of how to create a workload config file for user-defined workloads, please see the **Workload Configuration** section of this document.

Basic workload configuration options are also available from the workload configuration page on the controller web console; please refer to the **Workload Configuration** section of this document to customize the XML file for maximum flexibility. (Note that the <controller-host> IP address 192.168.250.36 shown in the screen capture below should be replaced with the actual IP address of the controller node.)

The workload configuration page supports to define multiple same stages, and also it allows to insert delay between stages to help identify boundary.



To define bulk workloads, we can use 'advanced config for workloads' hyperlink on controller web console. Advanced config UI helps to automatically generate a very large number of different combinations of various input parameters such as object sizes, objects per container, number of containers, workers, read-write-delete ratios.

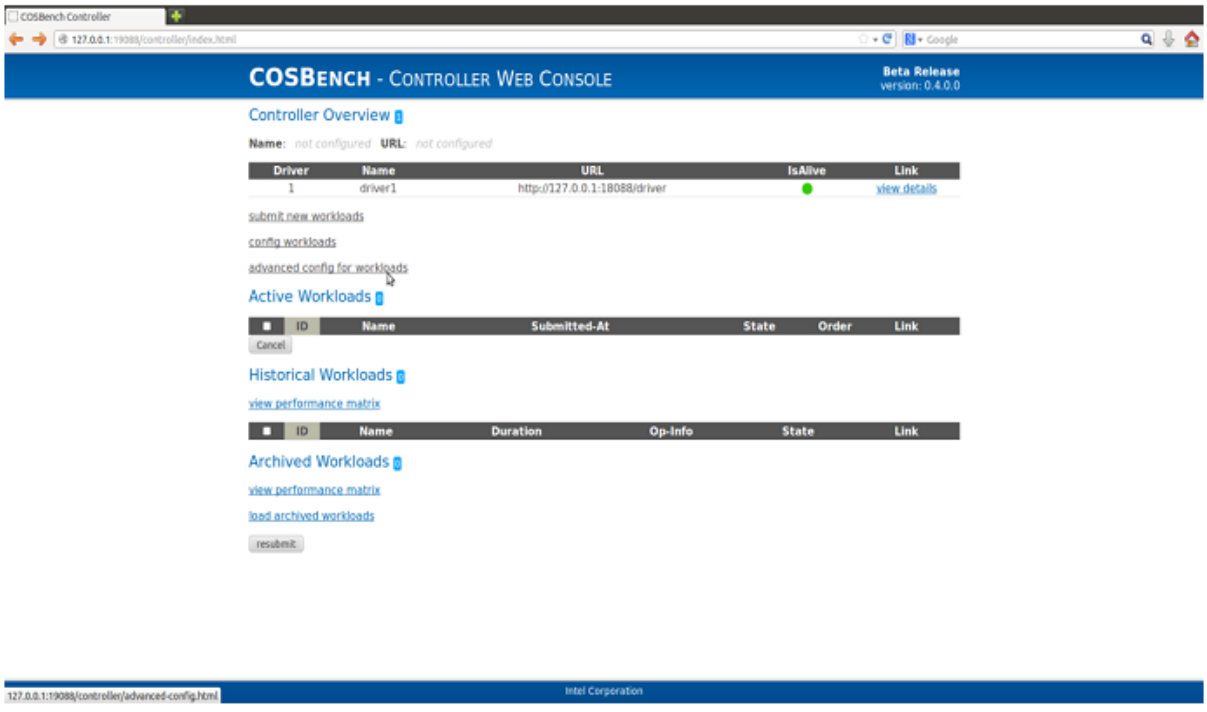
Once you click on 'advanced config for workloads' hyperlink, you will go to advanced config screen. From here, you can either generate workloads files or submit workloads directly.

In this section we will look into how to generate workload files. Whatever value you enter in 'Workload Matrix Name', a directory with that name will be generated inside 'workloads' directory under COSBench installation directory on machine where controller is installed.

For each workload you define from this screen, a file with name equal to the string you entered inside 'Workload Name' field will be created and will be placed under workload matrix directory created in previous step. There are some constraints on the names which you can enter in 'Workload Matrix Name' and 'Workload Name' fields. You should use alphabets or numbers, special characters allowed include _ - #. ( ) / % &. Length of string entered should be between 3 to 50 characters. Authentication and Storage configurations will be common to all workloads on advanced config UI page. Similarly attributes like

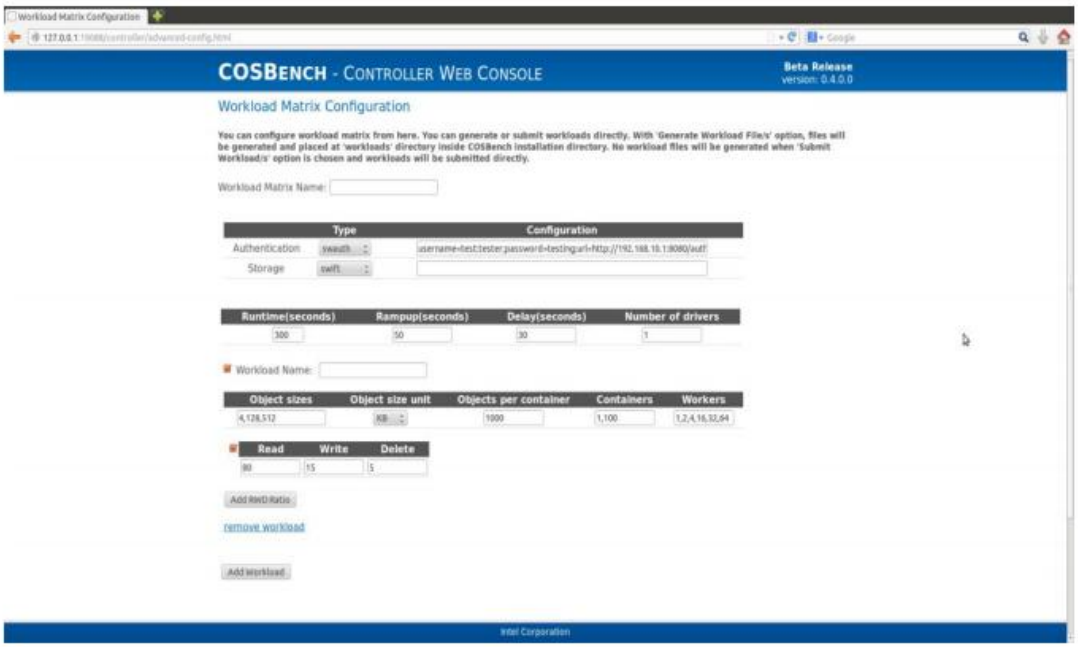
Runtime, Rampup, Delay and Number of drivers will be common to all workloads defined on this web page. Following input parameters are necessary for defining each workload:

Parameter	Type	Default	Comment
Object sizes	String	4,128,512	Can be comma separated or can be a range
Object size unit	String	KB	Drop down box consisting of values like Byte, KB, MB, GB
Objects per container	String	1000	Comma separated string
Containers	String	1,1000	Comma separated string
Workers	String	1,2,4,8,16,32,64	Comma separated string



Apart from these parameters, you can also add as many read-write-delete combinations as you want to any workload with the help of 'Add RWD ratio' button. You can also add as many workloads as you want with the help of 'Add Workload' button.

Once done with filling all these fields with appropriate values, you can then click on 'Generate Workload File/s' button. This will generate all configuration files at already mentioned location. You can edit these configuration files if you want and then submit them through workload submission UI screen. We also have option to submit to workloads directly through this page. We will look into that method in next sub-section.



3.6.2 Submitting Workloads

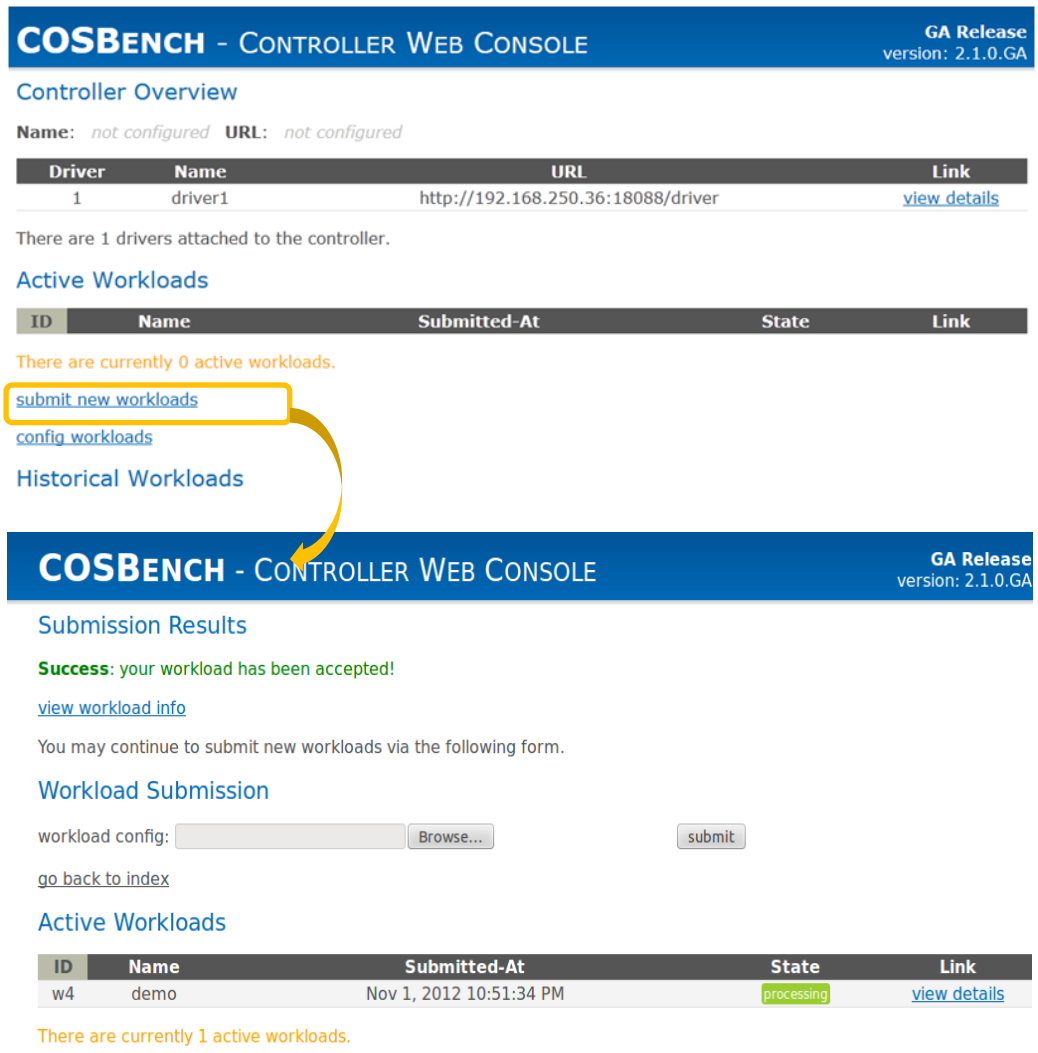
There are two ways to submit workloads to COSBench.

- Using the command-line interface:

  ```
  sh cli.sh submit conf/config.xml
  ```

- Using the web console:

  Open http://<controller-host>:19088/controller/index.html in a browser to monitor running status. (Note that the <controller-host> IP address 192.168.250.36 shown in the screen capture below should be replaced with the actual IP address of the controller node.)



3.6.3 Checking Workload Status

There are also two ways to check workload status.

- Using the command-line interface:

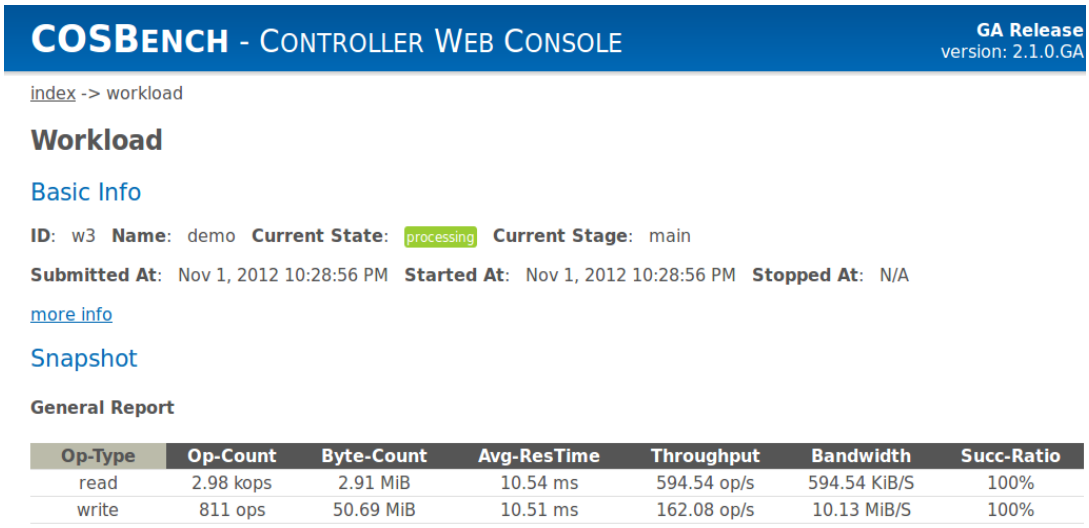
  ```
  sh cli.sh info
  ```

- Using the web console:

  Open http://<controller-host>:19088/controller/index.html in a browser to monitor running status. (Note that the <controller-host> IP address 192.168.250.36 shown in the screen capture below should be replaced with the actual IP address of the controller node.)

  You can also submit workloads directly through advanced config UI page. However, this page submits workloads generated which are defined through this page itself. You can use 'Submit Workload/s' button for the same. To learn how to define workloads through advanced config UI, please refer to previous sub-section.

Clicking **view details** in the Active Workload section of that interface screen displays runtime performance data, as shown below:



3.7 Stopping Drivers and Controllers

```
ps |grep java # you should see java here.
sh stop-driver.sh
ps |grep java # should be no java running.

ps |grep java
sh stop-controller.sh
ps |grep java
```

The “ps” command is used to help confirm whether the driver or controller process is stopped. If the Java process doesn’t stop as expected, the user may forcibly stop it by killing the process.

3.8 Configuring Tomcat

COSBench controller and driver use Apache Tomcat as the web server, the following table gives an overview of all the configurations related to Tomcat.

Configuration	Description	File Path
Tomcat for controller	Configuration for the web server on controller	conf/ controller-tomcat-server.xml
Tomcat for driver	Configuration for the web server on driver	conf/ driver-tomcat-server.xml
Tomcat web authentication	Configuration for web authentication, by default, there is a default username/password pair configured, user can change the “password” in configuration file to enforce username/password authentication when accessing web console.	conf/cosbench-users.xml

3.9 Workload management

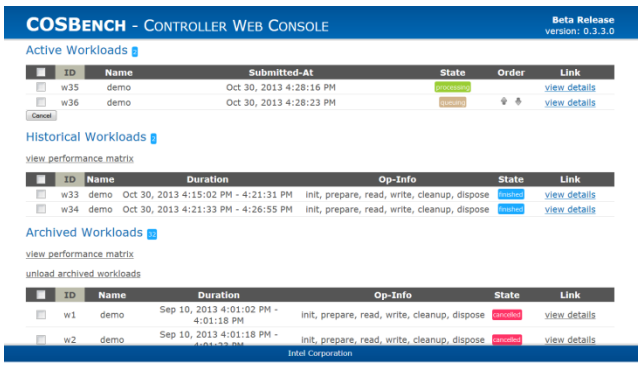
COSBench can accept multiple workload submissions, it maintains one job queue for those workloads, and executes them one by one.

On controller web console, workloads are organized into three sections:

- **Active workloads**: those are just submitted and not finished yet, including the one is in processing and those are in queue.
- **Historical workloads**: those are the workloads which have finished.
- **Archived workloads**: those are the workloads which were done in previous cosbench restart. COSBench can recognize workload results which were generated by previous instance, and can load or unload them on demand.

COSBench supports to manage those workloads through below approaches:

- **Reorder** workloads in active list
- **Load/unload** archived workloads
- **Re-submit** historic or archived workloads



4 Configuring Workloads

4.1 Introduction

```
workload
    auth
    storage
    workflow
        workstage
            auth
            storage
            work
                auth
                storage
                operation
```

A workload is represented as an XML file with the following structure:

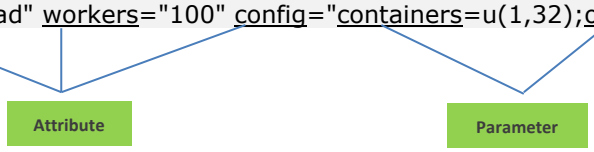
- Workload → work stage → work → operation
- If necessary, one workload can define one or more work stages.
- Execution of multiple work stages is sequential, while execution of work in the same work stage is parallel.
- For each piece of work, “workers” is used to tune the load.
- Authentication definition (auth) and storage definition (storage) can be defined at multiple levels, and lower-level definitions overwrite upper-level ones. For example, operations use the definitions for auth and storage at its work instead of those at workload level.

4.2 Selection Expression (also referred to as Selector)

4.2.1 Overview

- In workload configuration, the elements below support one “config” attribute (**auth**, **storage**, **work**, **operation**); the attribute contains an optional parameter list with key-value pairs that use the format “a=a_val;b=b_val”.

```
<work name="read" workers="100" config="containers=u(1,32);objects=u(1,100)" />
```



- In the parameter list, commonly used keys include "**containers**", "**objects**", and "**sizes**", which are used to specify how to select container, object, and size. One expression is used to help define selection.
- The number in an expression has a different meaning for object size versus object or container. For object size, the number represents a quantity, while for object or container, the number represents a numbering or label.

4.2.2 Selector

Expression	Format	Comments
constant	c(number)	Only use specified number For example, c(1) means the element numbering will be fixed in one fixed number
uniform	u(min, max)	Select from [min, max] evenly For example, u(1,100) means the element numbering is evenly selected from the 100 elements; the selection is random, and some numbers may be selected more than once, while some may never be selected
range	r(min,max)	Select from [min,max] incrementally For example, r(1,100) means the element numbering incrementally increases from min to max, and each number is selected only once; this is generally used in special stages (init, prepare, cleanup, dispose), if it's used in normal stage, please make sure you understand how to use it correctly (see FAQ 6.1.17 for details).
sequential	s(min,max)	Select from [min,max] incrementally For example, s(1,100) means the element numbering incrementally increases from min to max, and each number is selected only once. This is done thread-safe.

histogram	h(min1|max1|weight1,…)	It provides a weighted histogram generator, to configure it, specify a comma separated list of buckets where each bucket is defined by a range and an integer weight. For example: h(1|64|10,64|512|20,512|2048|30)KB Which defines one profile where (1,64)KB is weighted as 10, (64,512)KB is weighted as 20, and (512,2048)KB is weighted as 30. The sum of weights is not necessarily 100.

4.2.3 Allowable Combinations

There are additional constraints for selectors based on the element type and work type; the following two tables list allowable combinations.

Selector versus Element:

Key	constant (c(num))	uniform (u(min,max))	range (r(min,max))	sequential (s(min,max))	histogram (h(min|max|ratio))
containers	✓	✓	✓	✓	
objects	✓	✓	✓	✓	
sizes	✓	✓		✓	✓

Selector versus Work:

Key	init	prepare	normal (read)	normal (write)	normal (delete)	cleanup	dispose
containers	r(), s()	r(), s()	c(), u(), r(), s()	c(), u(), r(), s()	c(), u(), r(), s()	r(), s()	r(), s()
objects		r(), s()	c(), u(), r(), s()	c(), u(), r()	c(), u(), r(), s()	r(), s()	
sizes		c(), u(), h()		c(), u(), h()			

4.3 Workload

4.3.1 General Format

```
<workload name=”demo” description=”demo benchmark with mock storage” />
```

4.3.2 Attributes

Parameter	Type	Default	Comment
name	String		One name for the workload
description	String		Some additional information

4.4 Auth

4.4.1 General Format

```
<auth type="none|mock|swauth|keystone"
config="<key>=<value>;<key>=<value>" />
```

4.4.2 Attributes

Attribute	Type	Default	Comment
type	String	none	Authentication type
config	String		Parameter list [optional]

4.4.3 Authentication Mechanisms

- ✓ **none** (do nothing, default)

```
<auth type="none" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
logging	Boolean	false	Print information to log
retry	Int	0	Specifies number of retry attempts if authentication fails
Caching	Boolean	False	Caching authentication information or not

✓ **mock** (delay specified time)

```
<auth type="mock" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
token	String	“token”	Token string
delay	Long	20	Delay time in milliseconds
retry	Int	0	Specifies number of retry attempts if authentication fails

✓ **swauth** (for OpenStack Swift)

```
<auth type="swauth"
config="username=test:tester;password=testing;url=http://192.168.250.36:8080/auth/v1.0" />
```

Note that the IP address 192.168.250.36 should be replaced with the actual IP address of the controller node.

Parameter list:

Parameter	Type	Default	Comment
auth_url	String	“http:/192.168.250.36:8080/auth/v1.0”	URL for auth node
username	String		Username for authentication. Syntax h account:user
password	String		Password for authentication
timeout	Integer	30,000	Connection timeout value in milliseconds
retry	Int	0	Specifies number of retry attempts if authentication fails

✓ **keystone** (for OpenStack Swift)

```
<auth type="keystone"
config="username=tester;password=testing;tenant_name=test;url=http://192.168.250.36:5000/v2.0;service=swift" />
```

Note that the IP address 192.168.250.36 should be replaced with the actual IP address of the controller node.

Parameter list:

Parameter	Type	Default	Comment
auth_url	String	"http://192.168.250.36:8080/auth/v2.0"	URL for auth node
username	String		Username for authentication. Syntax account:user
password	String		Password for authentication
tenant_name	String		Name of tenant to which the user belongs
service	String	"swift"	Service requested
timeout	Integer	30,000	Connection timeout value in milliseconds
retry	Int	0	Specifies number of retry attempts if authentication fails

✓ **httpauth** (Http BASIC/DIGEST)

```
<auth type="httpauth"
config="username=test;password=testing;auth_url=http://192.168.250.36:8080/" />
```

Note that the IP address 192.168.250.36 should be replaced with the actual IP address of the controller node.

Parameter list:

Parameter	Type	Default	Comment
auth_url	String	“http://192.168.250.36:8080/”	URL for auth node
username	String		Username for authentication.
password	String		Password for authentication
timeout	Integer	30,000	Connection timeout value in milliseconds
retry	Int	0	Specifies number of retry attempts if authentication fails

4.5 Storage

4.5.1 General Format

```
<storage type="none|mock|swift|ampli|s3|sproxyd|..."
config="<key>=<value>;<key>=<value>" />
```

4.5.2 Attributes

Attribute	Type	Default	Comment
type	String	“none”	Storage type
config	String		Parameter list [optional]

4.5.3 Storage Systems

- ✓ **none** (do nothing, default)

```
<storage type="none" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
logging	Boolean	false	Print information to log

✓ **mock** (delay specified time)

```
<storage type="mock" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
logging	Boolean	false	Print information to log
size	Integer	1024	Object size in bytes
delay	Integer	10	Delay time in milliseconds
errors	Integer	0	Set error limit to emulate failure
printing	Boolean	False	Print out data content

✓ **Swift** (OpenStack Swift)

```
<storage type="swift" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
timeout	Integer	30,000	Connection timeout value in milliseconds
token	String	AUTH_xxx	Authentication token, this parameter is only necessary if user expects to bypass authentication.
storage_url	String	http://127.0.0.1:8080/auth/v1.0	The storage url, this parameter is only necessary if user expects to bypass authentication.

policy	String		Storage Policies is a feature introduced in Swift 2.0 that allows applications to select a different set of characteristics for their storage on a per container basis. See the latest Swift docs for full information at http://docs.openstack.org/developer/swift/overview_architecture.html. It's only necessary if user expects to leverage different storage policy instead of the default.

✓ **Ampli** (Amplidata)

```
<storage
type="ampli"config="host=192.168.10.1;port=8080;nsroot=/namespace;policy=14195ca863764fd48c281cb95c9bd555" />
```

Parameter list:

Parameter	Type	Default	Comment
timeout	Integer	30,000	Connection timeout value in milliseconds
host	String		Controller node IP to connect
port	Integer		Port
nsroot	String	"/namespace"	Namespace root
policy	String		Policy ID the namespace will access

✓ **S3 (Amazon S3)**

```
<storage type="s3" config="accesskey=<accesskey>;secretkey=<scretkey>;
endpoint=<endpoint>; proxyhost=<proxyhost>;proxyport=<proxyport>" />
```

Parameter list:

Parameter	Type	Default	Comment
timeout	Integer	30,000	Connection timeout value in milliseconds
accesskey	String		The base64-encoded username
secretkey	String		The base64-encoded password
endpoint	String	http://s3.amazonaws.com	The endpoint url (the url s3 storage exposes for external access).
proxyhost	String		The http proxy host name or ip address if required.
proxyport	integer		The http proxy port if required.

✓ Sproxyd (Scality)

```
<storage type="sproxyd" config="hosts=<host1,host2,…>;port=<port>;
base_path=<path>;pool_size=<maxTotal,maxPerRoute>" />
```

Parameter list:

Parameter	Type	Default	Comment
hosts	String	127.0.0.1	Comma separated list of host names/IP addresses. Requests are load balanced across all the hosts using a simple round robin algorithm
port	integer	81	Port used by the connector
base_path	String	/proxy/chord	Path to an sproxyd profile (this profile must have by_path_enabled = 1)
pool_size	integer or comma separated pair of integers	60,10	The first value s the size of the connection pool. The second value, if provided, is the maximum number of connections for a given HTTP route.

✓ Cdmi (SNIA CDMI)

```
<storage type="cdmi" config="type=<cdmi|non-cdmi;
custom_headers=<header:value_reference>" />
```

Parameter list:

Parameter	Type	Default	Comment
type	String	“cdmi”	Options: “cdmi” or “non-cdmi”, it indicates the content type to be used, “cdmi” means the storage access will follow cdmi content type, “non-cdmi” means the storage access will follow non-cdmi content type.
Customer_headers	String		This is an experimental parameter to see if possible to support cdmi derivatives, which may require additional headers. The parameter may be removed without notification.

✓ Cdmi_swift (SNIA CDMI for swift)

```
<storage type="cdmi_swift" config="" />
```

Parameter list:

Parameter	Type	Default	Comment
timeout	Integer	30,000	Connection timeout value in milliseconds

✓ librados (for Ceph)

```
<storage type="librados"
config="endpoint=<endpoint>;accesskey=<accesskey>;secretkey=<secretkey>" />
```

Parameter list:

Parameter	Type	Default	Comment
endpoint	String	127.0.0.1	The endpoint could be e.g. the monitor node.
accesskey	String		The username like “admin”.
secretkey	String		The secretkey is the key from the admin keyring.

Note:

- Don’t use librados to create containers (pools), they will default to only have 64 pgs, which renders them pretty useless, see http://ceph.com/docs/master/rados/operations/pools/

4.6 Work Stage

4.6.1 General Format

```
<workstage name="<name>" >
</workstage>
```

4.6.2 Attributes

Attribute	Type	Default	Comment
name	String		One name for the stage

4.7 Work

4.7.1 General Format

```
<work name="main" type="normal" workers="128" interval="5" division="none" runtime="60"
rampup="0" rampdown="0" totalOps="0" totalBytes="0" afr=”200000” config="" > . . . </work>
```

There is one normal and four special types of work (init, prepare, cleanup, and dispose). Section 4.7 focuses on normal work, while Section 4.8 covers the special types of work. The form given above is for a full set—different work types will have different valid forms. General rules are given below:

- *workers* is a key attribute, normally used to control load.
- *runtime* (including *rampup* and *rampdown*), *totalOps* and *totalBytes* are attributes that control how to end the work, called ending options. Only one can be set in a work.

4.7.2 Attributes

Attribute	Type	Default	Comment
name	String		One name for the work
type	String	“normal”	Type of work
workers	Integer		Number of workers to conduct the work in parallel
interval	Integer	5	Interval between performance snapshots
division	String	“none”	[“none” | “container” | “object”], controls how work is divided between workers
runtime	Integer	0	How many seconds the work will execute
rampup	Integer	0	How many seconds to ramp up workload; this time is excluded from runtime
rampdown	Integer	0	How many seconds to ramp down the workload; this time is excluded from runtime
totalOps	Integer	0	How many operations will execute; it should be a multiple of workers
totalBytes	Integer	0	How many bytes will transfer, it should be a multiple of the product of workers and size.

driver	string		Which driver will execute this work, by default, all drivers will participate the execution.
afr	Integer	200000 – normal 0 – special work	Acceptable failure rate, it's in millionth.

4.8 Special Work

4.8.1 General Format

```
<work type="init|prepare|cleanup|dispose|delay" workers="<number>"
config="<key>=<value>;<key>=<value>" />
```

Special work is different from normal work in the following ways:

- It internally adopts and calculates "totalOps", so *no ending option* need be explicitly included in the configuration.
- It has implicitly defined operations, so *no operation* is needed.
- "delay" is different from others, which causes the work just sleeps for specified seconds.

4.8.2 Supported Special Work

✓ **init** (creating specific *containers* in bulk)

```
<work type="init" workers="4" config="containers=r(1,100)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), r(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix

✓ **prepare** (inserting specific *objects* in bulk)

```
<work type="prepare" workers="4"
config="containers=r(1,10);objects=r(1,100);sizes=c(64)KB" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix
sizes	String		Size selection expression with unit (B/KB/MB/GB); for example: c(128)KB, u(2,10)MB
chunked	Boolean	False	Upload data in chunked mode (or not)
content	String	“random”(default) ”zero”	Fill object content with random data or all-zeros
createContainer	Boolean	False	Create related container if it does not exist
hashCheck	Boolean	False	Do work related to object-integrity checking

✓ **cleanup** (removing specific *objects* in bulk)

```
<work type="cleanup" workers="4" config="containers=r(1,10);objects=r(1,100)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix
deleteContainer	Boolean	False	Delete related container if it exists

✓ **dispose** (removing specific *containers* in bulk)

```
<work type="dispose" workers="4" config="containers=r(1,100)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix

✓ **delay** (inserting a few seconds delay)

```
<workstage name=”delay” closuredelay=”60” >
        <work type="delay" workers="1" />
</workstage>
```

Parameter list:

Parameter	Type	Default	Comment
closuredelay	Integer		How long to delay in seconds.

4.9 Operation

4.9.1 General Format

```
<operation type="read|write|delete" ratio="<1-100>"
config="<key>=<value>;<key>=<value>" />
```

4.9.2 Attributes

Attribute	Type	Default	Comment
type	String		Operation type
ratio	Integer		
division	Integer		Division strategy for this operation
config	String		Parameter list

4.9.3 Supported operations

✓ container/object naming convention:

- By default, containers are named using the format "**mycontainers_**<n>", and objects are named using the format "**myobjects_**<n>", where <n> is a number defined by one selection expression in the parameter list.
- Container/object naming can be modified through cprefix/csuffix or oprefix/osuffix.

✓ **read**

```
<operation type="read" ratio="70" config="containers=c(1);objects=u(1,100)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix
hashCheck	Boolean	False	Do work related to object-integrity checking

✓ write

```
<operation type="write" ratio="20"
config="containers=c(2);objects=u(1,1000);sizes=c(2)MB" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix
sizes	String		Size selection expression with unit (B/KB/MB/GB); for example: c(128)KB, u(2,10)MB
chunked	Boolean	False	Upload data in chunked mode (or not)
content	String	“random”(default) “zero”	Fill object content with random data or all zeros
hashCheck	Boolean	False	Do work related to object-integrity checking

✓ filewrite

```
<operation type="filewrite" ratio="20"
config="containers=c(2);fileselection=s;files=/tmp/testfiles" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
fileselection	String		Which selector should be used only put selector identifier (e.g. s for sequential). *
files	String		Path to the folder containing the files to be uploaded. Path must exist
chunked	Boolean	False	Upload data in chunked mode (or not)
hashCheck	Boolean	False	Do work related to object-integrity checking

*) Objects are not read by filename. Java reads the files in the folder in a random way. Use Sequential selection to assure each object will be picked once, before the first object is picked a second time. Limit the amount of objects put by using totalOps or runtime in your work definition.

✓ delete

```
<operation type="delete" ratio="10" config="containers=c(2);objects=u(1,1000)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix

✓ list

```
<operation type="list" ratio="10" config="containers=c(2);objects=u(1,1000)" />
```

Parameter list:

Parameter	Type	Default	Comment
containers	String		Container selection expression; for example: c(1), u(1,100)
cprefix	String	mycontainers_	Container prefix
csuffix	String	<null>	Container suffix
objects	String		Object selection expression; for example: c(1), u(1,100)
oprefix	String	myobjects_	Object prefix
osuffix	String	<null>	Object suffix

4.9.4 Examples

- ✓ pure read

```
e.g: 100% read, 16 users, 300 seconds
<work name="100r16c30s" workers="16" runtime="300">
    <operation type="read" ratio="100" config="..." />
</work>
```

- ✓ pure write

```
e.g.: 100% write, 8 clients, 600 seconds
<work name="100w8c600s" workers="8" runtime="600">
    <operation type="write" ratio="100" config="..." />
</work>
```

- ✓ mixed operations

```
e.g.: 80% read, 20% write, 32 clients, 300 seconds
<work name="80r20w32c300s" workers="32" runtime="300">
    <operation type="read" ratio="80" config="..." />
   <operation type="write" ratio="20" config="..." />
</work>
```

4.10 Additional comments

4.10.1 Overview

A few parameters need additional emphasis to make user define exact workload, this section will cover them.

4.10.2 Division strategy

Division strategies are used to divide a work into multiple non-overlapping partitions which have smaller ranges of containers or objects, there strategies are supported: **container** (based), **object** (based), or **none**.

Different stage type has different default division strategy, for init/dispose, the default is “container”, for prepare/cleanup, the default is “object”, and for normal, the default is “none”.

We will use one example to explain the difference between different division strategies, here is a work as following:

```
<work name="main" workers="4" runtime="300" division=”?”>
    <operation type="read" ratio="100" config="containers=u(1,8);objects=u(1,1000)" />
</work>
```

If "division=container", it means the data range will be partitioned by container, the access pattern looks like:

Worker	Container Range	Object Range
#1	1-2	1-1000
#2	3-4	1-1000
#3	5-6	1-1000
#4	7-8	1-1000

(Note: it's not supported if # of workers is larger than # of containers.)

If "division=object", it means the data range will be partitioned by object, the access pattern looks like:

Worker	Container Range	Object Range
#1	1-8	1-250
#2	1-8	251-500
#3	1-8	501-750
#4	1-8	751-1000

(Note: it's not supported if the # of workers is larger than the # of objects.)

If "division=none", it is used to turn off division so that each worker does exactly what the work has specified—there is no partitions of the work, so each worker may touch all containers or objects.

5 Results

All results are stored in the “archive” directory.

```
cosbench@cosbox:~/cos/archive$ ll
total 24
drwxrwxr-x 3 cosbench cosbench 4096 Oct 31 04:53 ./
drwxrwxr-x 8 cosbench cosbench 4096 Oct 31 04:53 ../
-rw-rw-r-- 1 cosbench cosbench    1 Oct 31 04:53 .meta
-rw-rw-r-- 1 cosbench cosbench  175 Oct 31 04:53 run-history.csv
drwxrwxr-x 2 cosbench cosbench 4096 Oct 31 04:53 w1-demo/
-rw-rw-r-- 1 cosbench cosbench  446 Oct 31 04:53 workloads.csv
```

5.1 Structure

- .meta
  - The starting run id
- run-history.csv
  - Record all historical workload runs, including time and major stages
- workload.csv
  - Record overall performance data for all historical workload runs
- Sub-directories
  - Prefixed with “w<runid>-” store data for each workload run

5.2 Per-Run Data

The following is a sample per-run data list:

```
cosbench@cosbox:~/cos/archive/w1-demo$ ll
total 1336
drwxrwxr-x 2 cosbench cosbench    4096 Oct 31 04:53 ./
drwxrwxr-x 5 cosbench cosbench    4096 Oct 31 23:37 ../
-rw-rw-r-- 1 cosbench cosbench     189 Oct 31 04:53 s1-init.csv
-rw-rw-r-- 1 cosbench cosbench     380 Oct 31 04:53 s2-prepare.csv
-rw-rw-r-- 1 cosbench cosbench    8035 Oct 31 04:53 s3-main.csv
-rw-rw-r-- 1 cosbench cosbench       0 Oct 31 04:53 s4-cleanup.csv
-rw-rw-r-- 1 cosbench cosbench       0 Oct 31 04:53 s5-dispose.csv
-rw-rw-r-- 1 cosbench cosbench     271 Oct 31 04:53 w1-demo.csv
-rw-rw-r-- 1 cosbench cosbench 1327852 Oct 31 04:53 w1-demo-rt-histogram.csv
-rw-rw-r-- 1 cosbench cosbench    3183 Oct 31 04:53 workload-config.xml
-rw-rw-r-- 1 cosbench cosbench    1346 Oct 31 04:53 workload.log
```

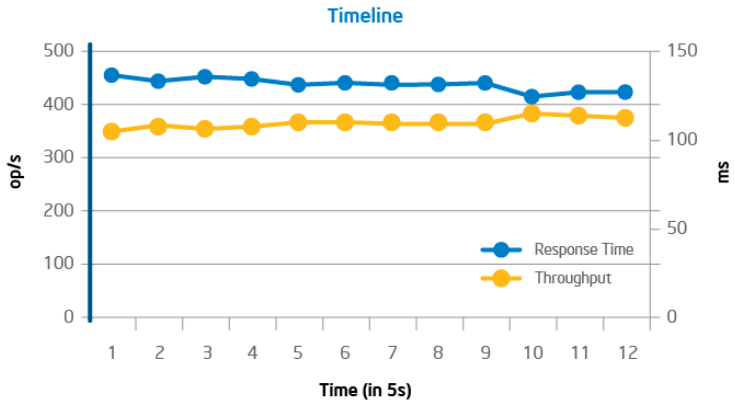
5.2.1 Overall Performance Data (e.g., w1-demo.csv)

One line per stage:

Id	Op	RT	TH	BW	Succ%
W1-s1-1	write	24,839	0.16	172	100%

5.2.2 Timeline Data (e.g., s3-main.csv)

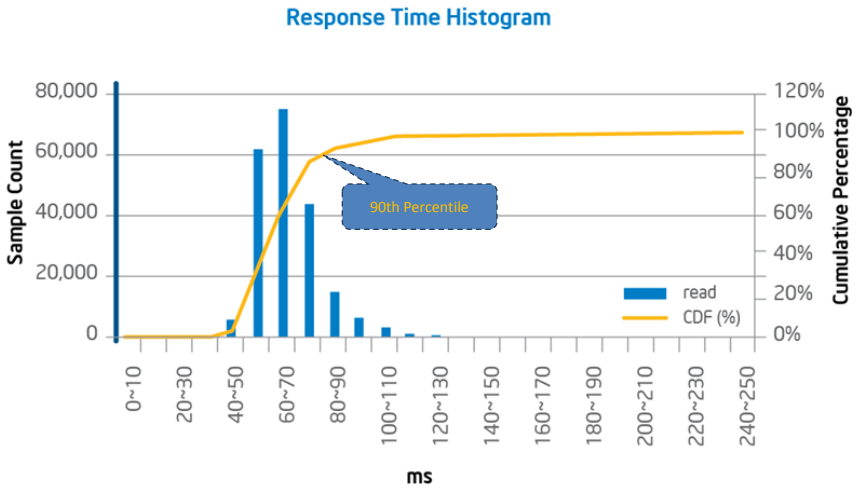
One file per stage; can be imported into a spreadsheet program to draw a timeline chart, or processed with csvtool:



5.2.3 Response-Time Histogram Data (e.g., w1-demo-rt-histogram.csv)

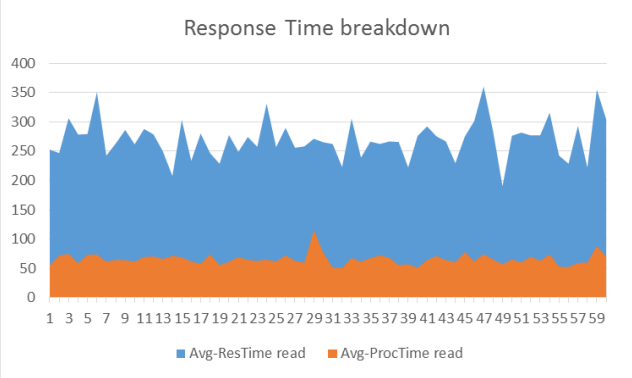
Distribution of response time is a valuable indicator to understand Quality of Service; histogram data is generated for this purpose. The data is grouped from 0 to 500,000 ms with 10 ms stepping.

In a histogram diagram, the bar represents the number of samples in each grouping. The curve is the Cumulative Distribution Function (CDF), which can reveal insights regarding topics such as the response time at the 90th percentile.



5.2.4 Response Time breakdown (e.g., s3-main.csv)

Beside average response time, average processing time will also be reported in data file, it would help understand performance bottleneck through time breakdown.



5.2.5 Workload-config.xml

- The workload configuration file used in this run

5.2.6 Workload.log

- The run time log, which is helpful for troubleshooting

5.3 Metrics

5.3.1 Throughput (Operations/s or Op/s)

- The operations completed in one second
- The reported throughput is calculated by dividing total successful requests by total run time

5.3.2 Response Time (in ms)

- The duration between operation initiation and completion
- The reported Response Time is the average of response time for each successful request
- One additional processing time is already reported to help breakdown response time.

5.3.3 Bandwidth (MB/s)

- The total data in MB transferred per second
- The reported bandwidth is calculated by dividing total bytes transferred by total run time
- 1 MB = 1000*1000 bytes

5.3.4 Success Ratio (%)

- The ratio of successful operations
- The reported success ratio is calculated by dividing the number of successful requests by the total number of requests

5.3.5 Other Metrics

- Op-count: total number of operations
- Byte-count: total data transferred

6 FAQs

6.1 General

1. **Is listening on port 19088/18088 configurable, and, if so, how?**

   Yes; conf/controller-tomcat-server.xml specifies the port to be used for the controller, and driver-tomcat-server.xml specifies the port to be used for the driver.

2. **What is the difference between "cancelled" and "terminated"?**

   "Cancelled" means the workload is cancelled by user at runtime, while "terminated" indicates errors during runtime, which typically require user action for resolution.

3. **Can I submit multiple workloads to be run sequentially?**

   Yes; COSBench can accept multiple workloads at one time and run them one by one.

4. **Is it possible to cancel a queued workload?**

   No; cancellation is only for the running workload.

5. **Can COSBench be installed on other Linux* distributions, such as Red Hat Enterprise Linux?**

   Yes; versions prior to v2.1 support Red Hat Enterprise Linux 6 by default, and versions beginning with v2.1 have adopted Ubuntu 12.04.1 LTS as the default OS.

6. **Is it possible to reuse the files from a previous test without removing or cleaning up the old files?**

   Yes; the special stages such as init, prepare, cleanup, and dispose are all optional, and even regarding the main stage, users can choose the stage sequence appropriate to their testing requirements. To reuse data, the user needs to fill all data and perform all tests before the cleanup and dispose stages (for example, in the sequence init, prepare, test1, test2, … cleanup, dispose). A related sample workload configuration file is included in conf/reusedata.xml.

7. **Is it possible to define multiple main or other stages?**

   Yes; to avoid name confusion, they should be named with different labels. For example, users can define multiple init stages to create different container sets, or define multiple main stages to perform different tests in one workload.

8. **If errors occur on running workloads, where can users see more details?**

   There is one workload.log under that workload's corresponding folder (archive\<workload id>\workload.log); inspecting this file can help determine the cause of errors.

9. **What steps should be taken to resolve a test being stuck at the init stage?**

   Verify that all COSBench machines are accessible through an HTTP connection using Curl ("curl http://<controller-host>:19088/controller/index.html" or "curl http://<driver-

host>:18088/driver/index.html"). If a firewall has blocked the HTTP connection, the user must open the appropriate ports on the firewall. For the controller node, the ports are 19088 and 19089; for driver nodes, the ports are 18088 and 18089.

10. **Is there a tool to distribute COSBench on multiple nodes?**

    Although COSBench itself does not provide a tool for this type of package distribution, many external solutions exist for this purpose, such as scp and shared folder (samba).

11. **Why does COSBench show a workload test as "complete," even though there are errors reported in workload.log?**

    A test may reflect "complete" status although errors are recorded in the log for normal work, as long as special work has completed successfully.

    COSBench treats "init", "prepare", "cleanup," and "dispose" operations as special work that must be completed without error to result in "completed" status; errors in special work will terminate the test.

    On the other hand, normal work associated with performance measurement can tolerate failures, which are tracked by the "success ratio."

12. **Are there any recommendations for the number of workers in the "init", "prepare", "cleanup," and "dispose" stages?**

    Work performed in the "init" and "dispose" stages creates and deletes containers. In our testing with Keystone plus Swift, these tasks can be completed in approximately three minutes with a recommended ratio of one worker for every 32 containers, with 100 objects in each container and a 64 KB object size. Generally, the number of containers should be defined as a multiple of the number of workers.

    Work performed in the "prepare" and "cleanup" stages creates or deletes objects, and the time required depends on the number of objects. Generally, the number of objects should be defined as a multiple of the number of workers. Increasing the number of workers can accelerate the process.

    Work performed in the "main" identifies bottlenecks, and tuning the workers parameter controls the load to the storage system. The number of workers should be gradually increased until performance decreases.

13. **How can "OutOfMemory" errors from the driver be prevented after running COSBench for a long time?**

    Maximum heap size for the Java process can be specified in the "cosbench-start.sh" script to prevent exhausting memory. For example, the parameter "-Xmx2g" would limit the maximum heap size to 2 GB.

14. **How can read and write be split to different containers?**

    Users can assign containers to be accessed at the operation level, to split reads and writes to different containers; the different container range can be set using the "containers" parameter in "config" as follows:

```
<operation type="read" ratio="80" config="containers=u(1,2);objects=u(1,50)" />
<operation type="write" ratio="20" config="containers=u(3,4);objects=u(51,100);sizes=c(64)KB" />
```

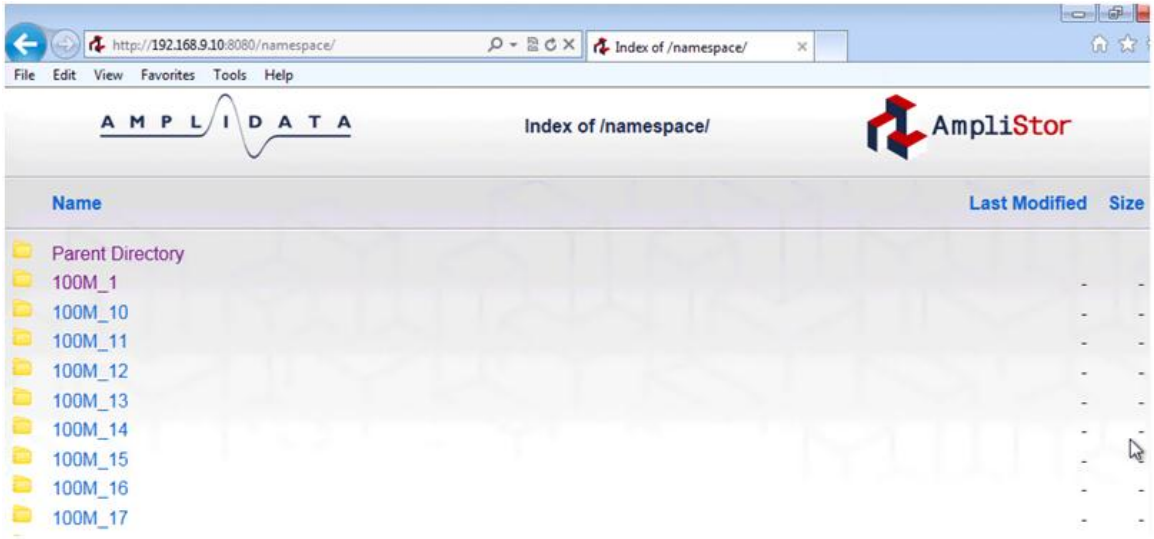
One sample workload configuration file is included in conf/splitrw.xml.

15. **How can different containers be specified for different configurations, such as those with different object sizes?**

    Users can assign different container sets for different configurations using the "cprefix" parameter. For example, users can differentiate between configurations with different sizes by specifying an object size such as "64K" using "cprefix" to avoid confusion and unexpected overwriting as follows:

    ```
    <operation type="read" ratio="80" config="cprefix=100M_;containers=u(1,32);objects=u(1,50)" />
    ```

    In this case, the container name will be prefixed with "100M_" in the target storage system. Users can take advantage of this capability by browsing to the location http://<IP of Amplistor controller node>:8080/namespace as shown below:



16. **When a workload is terminated, where can users obtain the log files to help troubleshoot the issue?**

    Log files are located in two separate places:

    - **In the "log" folder within the COSBench installation folder.** The "system.log" file in this location documents COSBench system activities, including the check for workload configuration file. If a required parameter is missing or mistyped (for example, "sizes" in the "prepare" stage), this file will contain an entry such as "driver report error: no such key defined: sizes" as shown below:

      ```
      2013-03-28 10:57:10,642 [ERROR] com.intel.cosbench.controller.tasklet.AbstractCommandTasklet - driver report error: no such
       key defined: sizes
      ```

    - **In the "workload" folder in the "archive" folder within the COSBench installation folder.** The "workload.log" file in this location documents workload runtime activities. If

failed operations occur while the workload is running, an error (typically HTTP-related) will be logged in this file.

17. **Can I use range selector in normal stage, how to do it?**

    Yes, range selector can be used in normal stage, but it's discouraged, as it will involve some subtle constraints. Firstly, range selector normally needs combine with "totalOps" to terminate the execution when all elements are enumerated. Secondly, the count of container should be relatively-prime to the count of objects, otherwise, the actual access range is only one of the greatest command divisor of the two counts. E.g., for below configuration, it's expected to create 1600 unique objects, but actually, it only creates 800 (=32*50/2) unique objects.

    ```
    <work name="main" workers="8" totalOps="1600">
        <operation type="read" ratio="100" config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
    </work>
    ```

18. **Can I run multiple drivers on the same node, how to do it?**

    Yes, COSBench doesn't constraint how many drivers can run on the same node, but as by default driver will listen on port 18088, to avoid port confliction, user needs set different driver process to different listening ports.

    To change the listening port, user needs edit conf/driver-tomcat-server.xml, and change "port" value below to distinguished number for different driver

    ```
    <Connector port="18088" protocol="HTTP/1.1" />
    ```

19. **If there's any way from COSBench side to identify which object was created failed ?**

    Yes, it's possible to print out the object name, but it requires to change the logging level for COSBench Driver processes. General steps are as following:

    i) On each driver node, creating one driver.conf file in conf/ folder with below lines if not exists:

    ```
    [driver]
    Log_level = DEBUG
    ```

    ii) Restarting all driver processes

6.2 Swift

1. **How can long prepare times associated with large object sizes (e.g., 1 GB) be avoided?**

   A large number of large objects can saturate network bandwidth, resulting in low performance. If network bandwidth is not saturated, the "workers" parameter can be increased, to better utilize bandwidth.

2. **Are there any special changes for operating with large object sizes (e.g., 1 GB)?**

   Yes; for testing with large object sizes, consider the following:

- Longer ramp-up time (specified using the parameter “rampup”) can help drive higher performance, while longer run time (specified using the parameter “runtime”) can help drive more consistent results. The optimal combination of settings for these parameters depends on individual usage circumstances. To help determine appropriate settings, run a test case with “runtime” set for a long run time (e.g., 30 minutes) without setting the “rampup” parameter. Consult the resulting timeline curve to determine how many seconds are needed to ramp up in this case; the “runtime” parameter can then be set to 10 times that “rampup” value.
- As the time for each operation to complete increases, it becomes more likely that timeout errors will occur; this effect can be mitigated using the “timeout” parameter in “config”, which uses milliseconds as units. For Swift, typical syntax is as follows:

  <storage type="swift" config="timeout=100000" />
- Users should also verify proper setup of the system under test, outside the scope of COSBench itself; for example, errors or performance deficits may occur because of improper setup of back-end storage.

3. **How can the termination of workloads in the authentication phase be overcome when large numbers of workers (e.g., 1024) are configured with Keystone, so that testing can be completed successfully?**

   The new parameter “retry” is introduced for the “auth” section in the workload configuration file to help overcome failures in the authentication phase. Following is a sample configuration:

   ```
   <auth type="keystone"
   config="username=operator;password=intel2012;tenant_name=cosbench;auth_url=http://1
   0.10.9.100:5000/v2.0;service=swift;retry=10"/>
   ```

4. **How to test with tempauth?**

   Tempauth actually follows the same procedure as swauth, so just use swauth authentication mechanism for tempauth.

5. **Is storage policy supported, how to get it working?**

   Yes, the storage policy (http://docs.openstack.org/developer/swift/overview_architecture.html) was supported starting from v0.4.0.0 release.

   The supporting involves one new “policy” parameter as following:

   ```
   <storage type="swift" config="policy=gold " />
   ```

   Here “gold” could be replaced with any policy name user defines in “/etc/swift/swift.conf” as following:

   ```
   [storage-policy:0]
   name = gold
   default = yes

   [storage-policy:1]
   ```

```
name = silver
```

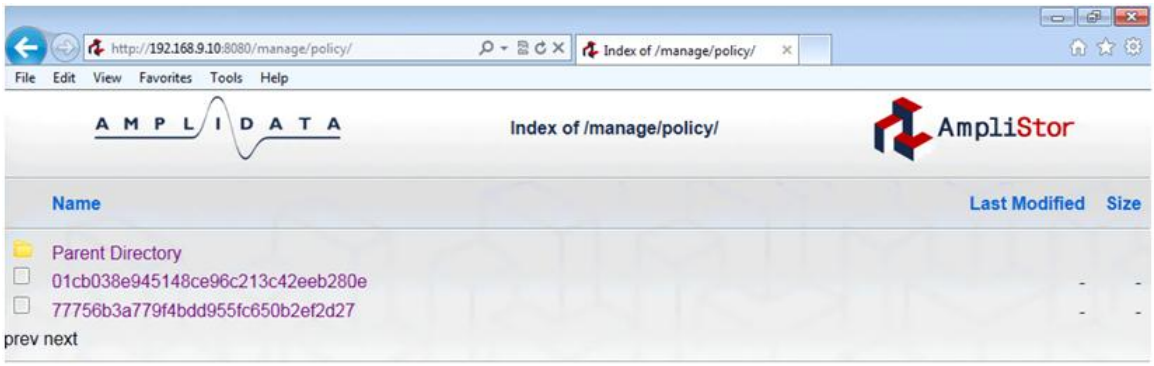
In this case, the policy name may be “gold” or “silver”, which will be used in COSBench workload configuration file.

For those who don’t care about storage policy, just remove “policy” parameter in swift workload configuration file.

6.3 AmpliStor

1. **Where does the system get the string for the policy in the .xml file?**

   Users can access the Amplidata controller node and manage the available policies by browsing to the location http://<IP of Amplistor controller node>:8080/manage/policy as shown below:



2. **How can object range affect performance?**

   Expanding the object range may improve write performance by reducing write conflicts. For example, changing “u(1,100)” to “u(1,10000)” will expand the object range from 100 objects to 10,000 objects.

3. **How can one simplify policy UID settings?**

   Only the “init” stage needs policy UID; other stages such as prepare, main, cleanup, and dispose don’t need to have the policy UID set. If there is no “init” stage in the workload, no policy UID is needed.

4. **How to use “nsroot” parameter in different stages?**

   “nsroot” parameter is introduced to support amplistor v2.5 and plus, where accessing namespaces needs a separate root path from objects. So this parameter is only necessary for the work involving namespace accessing like that in init/dispose stages. For other stages such as prepare, main, cleanup, there are two options, one is just removing “nsroot” parameter, another one is to set “nsroot” to **“/namespace”** instead of **“/manage/namespace”.**

   If “nsroot=/manage/namespace” is set in main stage, normally, some similar exceptions will pop up as below:

```
2013-11-20 10:45:33,764 [ERROR] [Writer] - fail to perform write operation
```

```
com.intel.cosbench.api.storage.StorageException:
org.apache.http.client.ClientProtocolException
...
Caused by: org.apache.http.client.NonRepeatableRequestException: Cannot retry
request with a non-repeatable request entity.  The cause lists the reason the
original request failed.
...
Caused by: java.net.SocketException: Broken pipe
...
```

6.4 S3

1. **What's the usage for parameter "proxyhost" and "proxyport"?**

   In some cases (e.g., in corporate network), users need go through one http proxy to reach Amazon S3 service, "proxyhost" and "proxyport" is used to give chance to configure http proxy settings.

2. **Can I route requests to specified region in Amazon S3?**

   S3 adaptor supports one parameter named "endpoint", which is capable to support routing requests to different regions. e.g., setting "**endpoint=https://s3-us-west-1.amazonaws.com**" will create buckets in Oregon region. Detailed s3 regions can be found at: http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region

6.5 Ceph

1. **What approaches are supported to access Ceph Object Storage?**

   COSBench supports to access Ceph Object Storage through Rados Gateway, in this case the exposed protocol for access could be S3 or Swift, depending on Rados Gateway configuration. From user's perspective, the Ceph cluster is a S3 or Swift compatible object storage, and the workload configuration follows S3 or Swift's configuration rules.

   Also, COSBench could interact with Ceph Object Storage with Rados protocol through librados. In this case, the storage adapter should be "librados". One Caveat is the ceph librados package should be installed on COSBench driver nodes.

6.6 CDMI

1. **How to test swift through cdmi protocol?**

   The storage type "cdmi_swift" is for swift, one sample workload file conf/cdmi-swift-config-sample.xml could help for reference.

2. **Why using two different cdmi storage types?**

   COSBench includes two cdmi related adapters: cdmi and cdmi-swift, the major difference is on authentication. CDMI standard adopts HTTP standard authentication mechanisms, while swift uses token-based authentication. CDMI is a general protocol to help accommodate different profiles in one standard, it's open for extensions, and it's expected to see more cdmi flavored adapters.

Appendix A. Sample Configurations

Swift

The sample workload configuration describes the following test scenario:

- The test includes five stages: init, prepare, main, cleanup, and dispose.
- The test creates 32 containers, each containing 50 objects 64 KB in size.
- The operation requests are issued to three controller nodes.
- The requests include 80 percent GET(read) operations and 20 percent PUT(write) operations; read operations randomly request an object from the 50 objects from #1 to #50, while write operations randomly create objects with object numbering from #51 to #100.
- At completion, the test cleans up all objects and drops all containers.

To use keystone authentication, use the commented keystone authentication line as a sample (note that the IP address 192.168.250.36 should be replaced with the actual IP address of the controller node).

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="swift-sample" description="sample benchmark for swift">

  <storage type="swift" />

  <!-- MODIFY ME -->
  <auth type="swauth"
config="username=test:tester;password=testing;auth_url=http://192.168.10.1:8080/auth/v1.0"
/>

  <!-- Keystone Authentication
  <auth type="keystone"
config="username=tester;password=testing;tenant_name=test;auth_url=http://192.168.250.36:5
000/v2.0;service=swift" />
  -->

  <workflow>

    <workstage name="init">
      <work type="init" workers="1" config="containers=r(1,32)" />
    </workstage>

    <workstage name="prepare">
      <work type="prepare" workers="1"
config="containers=r(1,32);objects=r(1,50);sizes=c(64)KB" />
    </workstage>

    <workstage name="main">
```

```
      <work name="main" workers="8" rampup="100" runtime="300">
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>
    </workstage>

    <workstage name="cleanup">
      <work type="cleanup" workers="1" config="containers=r(1,32);objects=r(1,100)" />
    </workstage>

    <workstage name="dispose">
      <work type="dispose" workers="1" config="containers=r(1,32)" />
    </workstage>

  </workflow>

</workload>
```

AmpliStor

The workload configuration describes the following test scenario:

- The test includes five stages: init, prepare, main, cleanup, and dispose.
- The test creates 32 containers (namespaces), each containing 50 objects 64 KB in size.
- The operation requests are issued to three controller nodes, and each controller node hosts two client daemons.
- The requests include 80 percent GET(read) operations and 20 percent PUT(write) operations; read operations randomly request objects from the 50 objects numbered #1 to #50, while write operations randomly create objects with object numbering from #51 to #100.
- At completion, the test cleans up all objects and drops all containers (namespaces).

For the AmpliStor v2.5 release, "*nsroot=/manage/namespace*" is necessary for all namespace-related work (init/dispose), for release prior to v2.5, just remove below "nsroot=/manage/namespace" snippets.

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="ampli-sample" description="sample benchmark for amplistor">

  <storage type="ampli"
config="host=192.168.10.1;port=8080;policy=14195ca863764fd48c281cb95c9bd555" />

  <workflow>

    <workstage name="init">
     <storage type="ampli"
config="host=192.168.10.1;port=8080;nsroot=/manage/namespace;policy=14195ca863764fd48c281c
b95c9bd555" />
```

```
     <work type="init" workers="1" config="containers=r(1,32)" />
    </workstage>

    <workstage name="prepare">
      <work type="prepare" workers="1"
config="containers=r(1,32);objects=r(1,50);sizes=c(64)KB" />
    </workstage>

    <workstage name="main">
      <work name="c1p0" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.1;port=8080" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>
      <work name="c1p1" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.1;port=8081" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>



      <work name="c2p0" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.2;port=8080" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>
      <work name="c2p1" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.2;port=8081" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>

      <work name="c3p0" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.3;port=8080" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>
      <work name="c3p1" workers="16" rampup="100" runtime="300">
        <storage type="ampli" config="host=192.168.10.3;port=8081" />
        <operation type="read" ratio="80" config="containers=u(1,32);objects=u(1,50)" />
        <operation type="write" ratio="20"
config="containers=u(1,32);objects=u(51,100);sizes=c(64)KB" />
      </work>
```

```
    </workstage>

    <workstage name="cleanup">
      <work type="cleanup" workers="1" config="containers=r(1,32);objects=r(1,100)" />
    </workstage>

    <workstage name="dispose">
    <storage type="ampli" config="host=192.168.10.1;port=8080;nsroot=/manage/namespace" />
      <work type="dispose" workers="1" config="containers=r(1,32)" />
    </workstage>

  </workflow>

</workload>
```

As mentioned at the beginning of this guide, COSBench also allows users to create adaptors for additional storage systems. Please refer to the “COSBench Adaptor Development Guide” for details.