OSDfailure Usage

This document provides step by step procedures for usage of the OSDfailure automation (github repo). It is intended to help new users understand the workflow and document the specific steps for using OSDfailure.

Introduction

OSDfailure was developed to automate performance testing of Ceph OSD failures while running an RGW access type workload. It uses the COSbench I/O workload generator (github repo) to drive RGW traffic into a pre-existing Ceph cluster.
The OSDfailure automation follows this workflow:

1. Edit 'vars.shinc' for your environment (hostnames; runtime; obj sizes ...)
2. run 'writeXMLs.sh' <-- create COSbench workload files
3. run 'prepCluster.sh' <-- create pools, add RGW user and fill the cluster
4. run 'runtest.sh' <-- run the test and record results

The flow of a test run (runtest.sh) is comprised of these three phases, all of which run for an equal amount of time:

- Phase 1: no failures
- Phase 2: OSD device failure
- Phase 3: OSD node failure

NOTE that while this automation is targeted for driving COSbench workloads, the failure injection automation can also be used with other I/O workload utilities. The runtest.sh script (line 72) invokes the I/O workload:

```
pbench-user-benchmark "Utils/cos.sh ${myPath}/${RUNTESTxml} $LOGFILE" &
```

That command line can be replaced with other I/O workload generators and the failure injection automation will be re-usable.

Pre-requisites

- Ceph cluster
- ADMIN node with:
    - password-less SSH into Ceph cluster (MONs, OSDs, RGWs)
    - Ceph client keyring privileges
- COSbench installed and configured (or edit ‘runtest.sh’ line 72)
- Pbench installed and configured (or edit ‘runtest.sh’ line 72)

Installation

```
# git clone https://github.com/jharriga/OSDfailure
# cd OSDfailure
# chmod 755 *.sh Utils/*.sh
```

Configuration (edits to ‘vars.shinc’)

Test Environment variables: *These are used by ‘prepCluster.sh’ to prepare the cluster and ‘runtest.sh’ to run the test*

- Rados Gateway URL: `rgwURL`
- Hostnames: `OSDhostname, MONhostname, RGWhostname`
- Ceph OSD cluster network interfaces: `IFACE_arr`
- Pool definition: `numREPLICAS, preparePTYPE, pg_data, pg_index, pg`
- Location of COSbench installation directory: `cosPATH`

I/O Workload variables: *These are used by ‘writeXML.sh’ to create the two COSbench workload files (PREPARExml, RUNTESTxml), from the existing templates (PREPAREtemplate, RUNTESTtemplate)*

- Runtimes: `failuretime, recoverytime`
- Object size range: `objSIZES`
- Number of Containers and Objects to prepare and test: `numCONT, numOBJ`
- Ratios for operation types: `rdRatio, wrRatio, delRatio, listRatio`
- Number of Workers: `PREPAREworkers; RUNTESTworkers`

Other configuration considerations:

- Ensure /etc/ansible/hosts has correct [rgws] entries (‘prepCluster.sh’ lines 66, 77)

Tuning the level of the I/O Workload

Determine RUNTESTworkers value

I/O Saturation (latency of 500 ms)

Production level (80% of saturation)

Execution

After the edits are made to ‘vars.shinc’, the commands should be run as root in this order:

```
[root@b10-h01-r620 OSDfailure]# ./writeXML.sh
[root@b10-h01-r620 OSDfailure]# ./prepCluster.sh
[root@b10-h01-r620 OSDfailure]# ./runtest.sh
```

Example outputs from these commands are provided in Appendix A

Note: ‘runtest.sh’ will generate a logfile (variable `LOGFILE` in ‘vars.shinc’) which records test phase steps, as well as ceph capacity and recovery statistics. By default this logfile will be written into the “OSDfailure/RESULTS” directory with a timestamped filename. The logfile, along with the COSbench and Pbench directories, will be copied to a uniquely named and timestamped directory under ‘/var/www/html/pub’. For the specific commands employed to perform these copy operations see ‘runtest.sh’ lines 194 - 198 and 234.

Variables and Order of Execution

The variables in the ‘vars.shinc’ file are used by all three scripts (writeXML.sh, prepCluster.sh and runtest.sh). To avoid unexpected test runs you must use caution when changing variable values without going back to the first step (writeXML.sh).
The I/O Workload variables (see above) are used by writeXML.sh to generate the two XML workload files (prepWorkload.xml and ioWorkload.xml). Some of them are also used in runtest.sh to coordinate timing of failure injections. If any of those variables are changed without regenerating the XML files then the XML workload files will no longer be aligned with the runtest.sh settings.
Let’s consider this scenario. The user executes writeXML.sh and then prepCluster.sh. Prior to executing runtest.sh they change the value of either the ‘failuretime’ OR ‘recoverytime’ variables. When they execute runtest.sh, the event timings will be incorrect - since the runtime values in the ioWorkload.xml file were generated from the old/original values for ‘failuretime’ and ‘recoverytime’. Similarly if the ‘numOBJ’ OR ‘numCONT’ values are changed after prepCluster.sh has been run, then the number of objects, and containers, will not match for the prepWorkload.xml and ioWorkload.xml files. This can lead to problems in runtest.sh, which executes the ioWorkload.xml file, trying to use uninitialized and unprepared objects or containers.

Appendix A: Example Command Output

writeXML.sh

```
[root@gprfc066 OSDfailure]# ./writeXML.sh
Creating COSbench XML workload files from settings in vars.shinc
> prepWorkload.xml exists - moved to prepWorkload.xml_bak
> created COSbench workload file: prepWorkload.xml
> ioWorkload.xml exists - moved to ioWorkload.xml_bak
> created COSbench workload file: ioWorkload.xml
DONE - Validate XML files before proceeding.
```

prepCluster.sh

```
[root@gprfc066 OSDfailure]# ./prepCluster.sh
prepCluster.sh: Running with these values:
```

```
RGWhostname=smerf02 r=rep k=0 m=0 pgdata=2048 pgindex=64        pg=64 f=0
Stopping RGWs
smerf02 | SUCCESS | rc=0 >>

smerf01 | SUCCESS | rc=0 >>

Removing existing/old pools
pool 'default.rgw.users.keys' removed
pool 'default.rgw.data.root' removed
pool '.rgw.root' removed
pool 'default.rgw.control' removed
pool 'default.rgw.gc' removed
pool 'default.rgw.buckets.data' removed
pool 'default.rgw.buckets.index' removed
pool 'default.rgw.buckets.extra' removed
pool 'default.rgw.log' removed
pool 'default.rgw.meta' removed
pool 'default.rgw.intent-log' removed
pool 'default.rgw.usage' removed
pool 'default.rgw.users' removed
pool 'default.rgw.users.email' removed
pool 'default.rgw.users.swift' removed
pool 'default.rgw.users.uid' removed
Creating new pools
pool 'default.rgw.users.keys' created
set pool 53 size to 2
pool 'default.rgw.data.root' created
set pool 54 size to 2
pool '.rgw.root' created
set pool 55 size to 2
pool 'default.rgw.control' created
set pool 56 size to 2
pool 'default.rgw.gc' created
set pool 57 size to 2
pool 'default.rgw.buckets.data' created
set pool 58 size to 2
pool 'default.rgw.buckets.index' created
set pool 59 size to 2
pool 'default.rgw.buckets.extra' created
set pool 60 size to 2
pool 'default.rgw.log' created
set pool 61 size to 2
pool 'default.rgw.meta' created
set pool 62 size to 2
pool 'default.rgw.intent-log' created
set pool 63 size to 2
pool 'default.rgw.usage' created
set pool 64 size to 2
pool 'default.rgw.users' created
set pool 65 size to 2
```

```
pool 'default.rgw.users.email' created
set pool 66 size to 2
pool 'default.rgw.users.swift' created
set pool 67 size to 2
pool 'default.rgw.users.uid' created
set pool 68 size to 2
enabled application 'rgw' on pool 'default.rgw.users.keys'
enabled application 'rgw' on pool 'default.rgw.data.root'
enabled application 'rgw' on pool '.rgw.root'
enabled application 'rgw' on pool 'default.rgw.control'
enabled application 'rgw' on pool 'default.rgw.gc'
enabled application 'rgw' on pool 'default.rgw.buckets.data'
enabled application 'rgw' on pool 'default.rgw.buckets.index'
enabled application 'rgw' on pool 'default.rgw.buckets.extra'
enabled application 'rgw' on pool 'default.rgw.log'
enabled application 'rgw' on pool 'default.rgw.meta'
enabled application 'rgw' on pool 'default.rgw.intent-log'
enabled application 'rgw' on pool 'default.rgw.usage'
enabled application 'rgw' on pool 'default.rgw.users'
enabled application 'rgw' on pool 'default.rgw.users.email'
enabled application 'rgw' on pool 'default.rgw.users.swift'
enabled application 'rgw' on pool 'default.rgw.users.uid'
sleeping for 400s seconds...
Starting RGWs
smerf01 | SUCCESS | rc=0 >>

smerf02 | SUCCESS | rc=0 >>

Creating User - which generates a new Password
{
    "user_id": "johndoe",
    "display_name": "John Doe",
    "email": "john@example.com",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
    {
           "user": "johndoe",
           "access_key": "64YT9IY5SDDVMBS7L206",
           "secret_key": "PBNKKSxAmLuUZLs7adOnYGMwkw1mmppAYwwFBIeJ"
    }
    ],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
```

```
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
    },
    "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}

{
    "user_id": "johndoe",
    "display_name": "John Doe",
    "email": "john@example.com",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
    {
           "id": "johndoe:swift",
           "permissions": "full-control"
    }
    ],
    "keys": [
    {
           "user": "johndoe",
           "access_key": "64YT9IY5SDDVMBS7L206",
           "secret_key": "PBNKKSxAmLuUZLs7adOnYGMwkw1mmppAYwwFBIeJ"
    }
    ],
    "swift_keys": [
    {
           "user": "johndoe:swift",
           "secret_key": "DrSQjIXNKbfehKPYBitL5sFjBjnfrz6zNQZZDKeD"
    }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
```

```
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
        },
        "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
        },
        "temp_url_keys": [],
        "type": "rgw"
    }

    inserting new password into XML files prepWorkload.xml, ioWorkload.xml
    starting the I/O workload to prepare the Ceph cluster
      % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                     Dload  Upload   Total   Spent    Left  Speed
    100  1191  100    21  100  1170       85   4738 --:--:-- --:--:-- --:--:--  4756
    2018/02/15:14:07:42: Accepted with ID: w64
    2018/02/15:14:07:42: COSbench jobID is: w64 - Started
```

runtest.sh

```
# ./runtest.sh
2018/02/23:10:33:41: runtest.sh - Created logfile:
./RESULTS/runtest.sh_20180223-103341.log
2018/02/23:10:33:42: > OSDhost is smerf02 : smerf02.sbu.lab.eng.bos.redhat.com
2018/02/23:10:33:43: > MONhost is gprfc072 : gprfc072.sbu.lab.eng.bos.redhat.com
2018/02/23:10:33:44:
GLOBAL:
        SIZE          AVAIL         RAW USED      %RAW USED
        87579G        74628G        12951G        14.79
POOLS:
        NAME                        ID     USED   %USED  MAX AVAIL     OBJECTS
        default.rgw.buckets.data    122    6408G         3.02          22830G
1900485
noscrub is set
nodeep-scrub is set
2018/02/23:10:33:46: ** pbench-user-benchmark cosbench started as PID: 31790
2018/02/23:10:33:48: BEGIN: No Failures - start sleeping 10m
Running Utils/cos.sh ./ioWorkload.xml ./RESULTS/runtest.sh_20180223-103341.log
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                     Dload  Upload   Total   Spent    Left  Speed
100  2030  100      21  100  2009       85   8204 --:--:-- --:--:-- --:--:--  8233
2018/02/23:10:34:02: Accepted with ID: w96
2018/02/23:10:34:02: COSbench jobID is: w96 - Started
2018/02/23:10:43:48: CONTINUE: No Failures - start sleeping 60m
2018/02/23:11:43:48: END: No Failures - completed sleeping
```