CeTune Document

Revision <2.1>

Last Print Date: 8/13/2015 - 10:54 AM

Primary Author(s): **Chendi, Xue Chendi.xue@intel.com**

1. Table of Contents

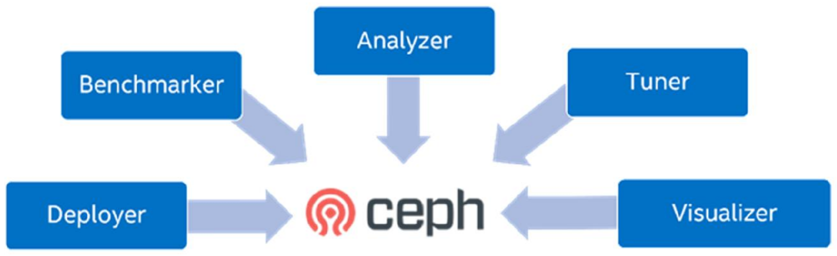
1.Introduction of CeTune

1.1 What is CeTune

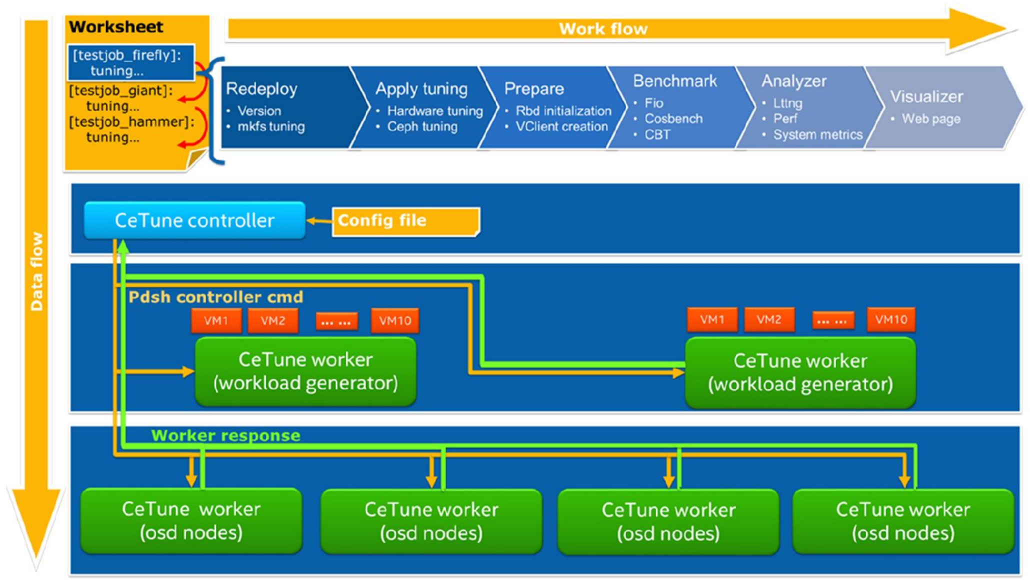
CeTune is a python-based framework, designed and implemented to profile and tune ceph cluster performance. This is the v1 version of CeTune, which is rewrite from the original shell-based Automation kit (cephperf).

In CeTune, we designed it to be kicked off by one click from deploy ceph cluster to benchmark and profile ceph and show performance report with HTML.

To reach that goal, CeTune comprises five distinct components. And using two config files to drive all these five components.



There are two roles in CeTune, CeTune controller and CeTune worker. Each ceph node includes mon, osd and client are all CeTune worker. They will receive job from CeTune controller to do deployment, create and attach rbd, run workload, recording system metrics and etc. The whole workflow overview is like below:

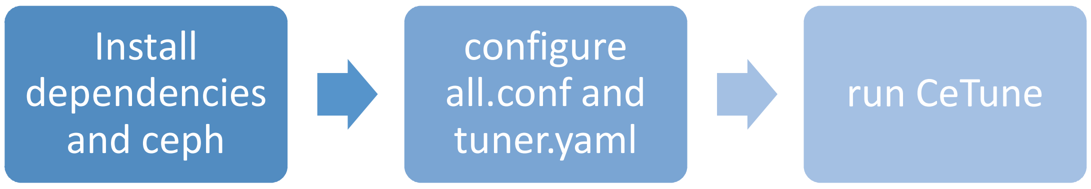


CeTune controller node can be any node in ceph cluster or even an individual node. We recommend to put CeTune controller at one of your client node. (Our test is on this scenario, so this will be more stable and bugless.)

1.2 How to use CeTune

CeTune is being able to help engineers doing deployment, benchmarking, vm setting up and other works like autossh, disk_partition.

We aim to make using CeTune as simple as possible, with only three major steps, you can run CeTune to do a whole test from zero ceph in cluster to complete the ceph performance report.


Install dependencies and ceph
configure all.conf and tuner.yaml
run CeTune

2. Preparations (install dependencies and ceph)

2.1 Install ceph-deploy

Ceph-deploy will be called from CeTune scripts to install and deploy ceph, so before using CeTune, make sure the ceph-deploy is installed.

```
[CeTune_controlle]
pip install ceph-deploy
#check ceph-deploy is installed, the CeTune is tested from the ceph-deploy version of 1.5.2,
so please install a version higher than that, avoiding any unknown problems may occur.
ceph-deploy --version
```

2.2 Install dependencies

We install ceph with ceph-deploy, which requires network.

```
[CeTune_controlle]
apt-get/yum install pdsh
apt-get/yum install unzip
apt-get/yum install python-lxml
apt-get/yum install python-yaml
apt-get/yum install python-argparse
apt-get/yum install python-matplotlib
apt-get/yum install python-numpy
apt-get/yum install expect
apt-get/yum install sysstat
apt-get/yum install curl
apt-get/yum install openjdk-7-jre
apt-get/yum install haproxy
pip install ceph-deploy
```

3. Ceph installation and deployment

3.1 Ceph installation

Current CeTune installs ceph by ceph-deploy and only support installing rbd as client. If you prefer installing ceph from source, there is a quick manual in 3.1(b).

a) Check the network connection

Please edit the files /etc/apt/apt.conf, /etc/environment and /etc/wgetrc to make sure that the proxy of all node in your cluster are available, including client and server, and if your system is CentOS, you need to edit your file /etc/yum.conf to make sure your proxy is available.

```
[CeTune_worker]
Check apt-get / yum is able to connect to the repository
Wget ceph.com #make sure wget is able to connect to ceph.com
```

Add all Ceph node include osd, mon, client into [CeTune controller]'s hosts file

If your cluster using different nic driver as cluster network and public network, remember to add all nic driver ip into /etc/hosts.

```
[CeTune_controller]
make sure all your osd, mon, rbd client can be connected through the controller node.
Or you can use script in CeTune help you to do the autossh setting.

apt-get/yum install expect
Cd deploy/prepare-scripts; ./configure_autossh.sh ${host} ${ssh_password}
```

b) Config conf/all.conf file

```
[CeTune_controller]
vim conf/all.conf
conf_path="/etc/ceph/ceph.conf"
deploy_username=ceph
deploy_ceph_version=hammer
deploy_mon_servers=aceph01
deploy_osd_servers=aceph01,aceph02,aceph03,aceph04
deploy_rbd_nodes=client01,client02

bash deploy-ceph.sh install
```

c) Purge ceph

```
for i in ${all_nodes}; do ssh $i ceph -v; done
#if you want to remove current installed ceph, using command below
bash deploy-ceph.sh purge
or
enter your ceph dir to make uninstall
```

You can see ceph -v at the end if you succeeded in installing ceph

3.2 Ceph deployment

CeTune provides Three way to deploy ceph, by CeTune-redeploy, by ceph-deploy and by CBT.

If you are planning to do benchmark and profile ceph, we recommend using CeTune-redeploy.

Before deploy, check the partition of all osd device to make sure they are deployed correctly.

3.2.1 Prepare osd devices

```
[CeTune_controller]
vim conf/all.conf
#config each osd node with osd_device and journal_device
osd_node1=${osd_device}:${journal_device}
example:
aceph01=/dev/sda1:/dev/sdb1,/dev/sdd1:/dev/sdb2,/dev/sde1:/dev/sdb3…
# how many partitions on one osd device and the size
# CeTune has script to do partitioning
# set the osd_partition_size to " ", CeTune will use whole disk space as one partition
osd_partition_count=1
osd_partition_size=2000G
# how many partitions on one journal device and the size
journal_partition_count=5
journal_partition_size=60G

cd deploy/prepare-scripts;
bash list_disk_partition.sh –l  #To check current osd and journal device partition
bash list_disk_partition.sh –w  #apply all.conf to osd and journal device
```

3.2.2 Deploy ceph by CeTune-redeploy

If you plan to deploy ceph by CeTune-redeploy, you can skip this part, directly to 5.Run CeTune.

3.2.3 Deploy ceph by ceph-deploy

d) Deploy ceph monitor

```
bash deploy-ceph.sh deploy mon
# Monitor number should be an odd number
```

e) Deploy ceph mds/osd

```
# Before deploying osd, you can change the ceph.conf in deploy folder which is generated by
"ceph-deploy mon" for some cluster setting; or you can also set it in runtime
Bash deploy-ceph.sh deploy osd
```

f) Check ceph healthy

```
ceph -s
    cluster b9e48882-7485-4312-ac5b-4148409d59e1
    health HEALTH_OK
    monmap e1: 1 mons at {aceph01=192.168.5.11:6789/0}, election epoch 2, quorum 0 aceph01
    osdmap e134: 20 osds: 20 up, 20 in
     pgmap v2517: 480 pgs, 1 pools, 94864 MB data, 23727 objects
          95879 MB used, 18527 GB / 18621 GB avail
               480 active+clean

#You should see ceph is healthy now, if not, there could be some reasons below
    a) pg number is too big or too small
    ceph osd pool set {pool-name} pg_num {pg_num}
    ceph osd pool set {pool-name} pgp_num {pgp_num}

    b) replica number is larger then ceph cluster nodes number
    ceph osd pool set {pool-name} size {replica_size}
```

4. VM setup

If you plan to run qemufio benchmark on ceph cluster, CeTune also provides some scripts to setup VM, skip this chapter if you not plan to run fio in VM.

4.1 Set up hypervisor bridge network.

a) Edit /etc/network/interfaces file.

Add a new ‘br0’ network who bridge_ports to a physical nic (In the example, it points to ‘eth0’), and copy the original physical nic(eth0) address, network, gateway settings to br0. Example is like below, ‘eth0’ should be replaced by the nic can connect to network in your own system.

```
[Ubuntu]
auto eth0
iface eth0 inet manual

auto br0
iface br0 inet static
address 192.168.5.31
netmask 255.255.0.0
gateway 192.168.2.200
bridge_ports eth0
bridge_stp off
bridge_fd 0
bridge_maxwait 0

[CentOS]
[root@client03 ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
TYPE=Bridge
```

```
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.5.31
NETMASK=255.255.0.0
DELAY=0
GATEWAY=192.168.2.200
[root@client03 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:1e:67:92:3c:2d
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
USERCTL=no
```

b) Restart the network service.

```
[root@client01 ~]# ifdown eth0
[root@client01 ~]# ifup eth0
Possible Error:
Ignoring unknown interface br0=br0.
Solution:
[root@client01 ~]# apt-get/yum install bridge-utils
```

c) Check the network setting

```
# System route to 192.168.0.0 by br0
root@ceph-client1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.2.200   0.0.0.0         UG    0      0        0 br0
172.16.96.0     *               255.255.240.0   U     0      0        0 eth2
172.16.96.0     *               255.255.240.0   U     0      0        0 eth5
192.168.0.0     *               255.255.0.0     U     0      0        0 br0
192.168.122.0   *               255.255.255.0   U     0      0        0 virbr0
```

4.2 Prepare VMs

d) Use the scripts in vm-scripts to prepare vm

```
[CeTune_controller]
Set conf/all.conf to create vm image
vim conf/all.conf
list_vclient=vclient01,vclient02,vclient03,vclient04…
cpuset_start=0  #when pin vm vcpu to hypervisor cpu, will start from cpu 0
vm_num_per_client=40  #after create 40 vms, will re-pin vcpu to hypervisor cpu 0
img_path_dir=/mnt/images  #output created vm image folder
ip_prefix=192.168.5 #ip_prefix and ip_fix specify the vm ip,
ip_fix=161          #in below case, vm ip start from 192.168.5.161
vm_image_locate_server=10.239.158.45    #the remote_dir of tmp_vclient.image
```

```
cd vm-scripts/
bash prepare-vm.sh
scp –r vmxml/ ${img_path_dir_mnt} ${client_node}  #scp vclient.xml and vclient.img to clients
```

4.3 Create VMs

```
virsh create {xml-file-path}
```

Tip: check the file /etc/hosts in one of clients(the one which has dir CeTune), make sure it had recorded all IP addresses of VMs. After the creating of VMs, please guarantee that all the VMs can be logged on with SSH.

```
[example]
for i in `seq 1 4`; do virsh create vmxml/vclient${i}.xml;done
Domain vclient01 created from vmxml/vclient01.xml
Domain vclient02 created from vmxml/vclient02.xml
Domain vclient03 created from vmxml/vclient03.xml
Domain vclient04 created from vmxml/vclient04.xml

root@ceph-client1:~/CeTune/vm-scripts# virsh list
 Id    Name                           State
----------------------------------------------------
 19    vclient01                      running
 20    vclient02                      running
 21    vclient03                      running
 23    vclient04                      running
```

4.4 Rbd volume creation

If you choosing deploy ceph by CeTune-deploy, just skip this part.
Below is the instruction mostly used for deploy ceph with cephx authentication.

e) Create rbd volume

This script will read the current rbd volume number and vclient number in all.conf, then create proper number of rbd volumes to ensure each vm has one rbd volume. So if you has enough rbd volume for vclient, this script will do nothing.

```
bash create-volume.sh create_rbd
```

f) Create rbd_disk.xml

```
#create rbd volume xml with cephx authentication
bash create-volume.sh create_disk_xml
If you use CephX, pls make sure the secret.xml locates in vm-scripts
1) secret.xml exists, continue with cephx
2) help to generate secret.xml first than create volume
3) continue with none auth
#? 2
Secret b9e48882-7485-4312-ac5b-4148409d59e1 created
Secret value set

cat vdbs/vclient01.xml
<disk type='network' device='disk'>
   <driver name='qemu' type='raw' cache='none'/>
   <auth username='admin'>
      <secret type='ceph' uuid='b9e48882-7485-4312-ac5b-4148409d59e1'/>
   </auth>
   <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a' />
   <target dev='vdb' bus='virtio'/>
   <serial>009ad738-1a2e-4d9c-bf22-1993c8c67ade</serial>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</disk>

#create rbd volume xml with none authentication
bash create-volume.sh create_disk_xml
If you use CephX, pls make sure the secret.xml locates in vm-scripts
1) secret.xml exists, continue with cephx
2) help to generate secret.xml first than create volume
3) continue with none auth
#? 3

cat vdbs/vclient01.xml
<disk type='network' device='disk'>
   <driver name='qemu' type='raw' cache='none'/>
   <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a' />
   <target dev='vdb' bus='virtio'/>
   <serial>009ad738-1a2e-4d9c-bf22-1993c8c67ade</serial>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</disk>
```

g) rbd volume attach

```
virsh attach-device ${vclient} ${vclient.xml}
```

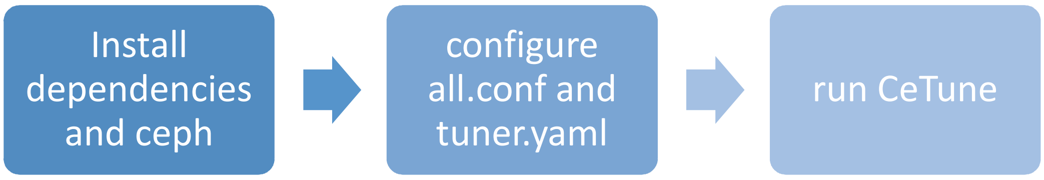
```
[example]
for i in `seq 1 4`; do virsh attach-device vclient0${i} vdbs/vclient${i}.xml; done
Device attached successfully
Device attached successfully
Device attached successfully
Device attached successfully

#check the rbd volume is attached to vm successfully
for i in `seq 1 4`; do echo vclient0${i}; virsh dumpxml vclient0${i} | grep rbd; done
vclient01
     <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a'/>
```

```
vclient02
     <source protocol='rbd' name='rbd/volume-a856d868-a065-4127-8b7b-c0d1dfb06e77'/>
vclient03
     <source protocol='rbd' name='rbd/volume-ad0c107a-8d8a-47eb-b558-23ad7ee61967'/>
vclient04
     <source protocol='rbd' name='rbd/volume-db060f87-08df-4c8c-ba6d-6f7a94d64531'/>
```

5. Benchmark ceph

5.1 Basic Execution Flow

As mentioned at chapter 2, the basic execution flow to use CeTune is
(1)install, (2)configure, (3)run.
Till now, we had installed ceph and all the dependencies; and create vclient if necessary.
All remained to do is do the configuration and kickoff CeTune.


Install dependencies and ceph
configure all.conf and tuner.yaml
run CeTune

5.2 Configuration

5.2.1 conf/all.conf

```
[CeTune_controller]
vim conf/all.conf
#====================BENCHMARK====================
head=ceph-client1 #CeTune controller node
tmp_dir=/opt
user=root
list_mon=aceph01
list_client=ceph-client1
list_ceph=aceph01,aceph02
volume_size=10240  #rbd volume size to create
rbd_volume_count=80   #how many rbd volume to create in total
benchmark_engine=cosbench,qemurbd,fiorbd   #define the workload engine
run_vm_num=80,70,60,50,40,30,20,10,4,2,1   #How many rbd volume will be tested.
run_file=/dev/vdb   #if using vclient, rbd will be attached as /dev/vdb
run_size=10g  #fio test size, don't exceed the rbd volume size
run_io_pattern=seqwrite,seqread,randwrite,randread
run_record_size=64k,4k
run_queue_depth=32,8,2
run_warmup_time=0
```

```
run_time=300
dest_dir=/mnt/data/
dest_dir_remote_bak=192.168.3.101:/data4/Chendi/ArborValley/v0.91/raw/  #The remote dir to
bak up the test result.
rbd_num_per_client=40,40   #if you have two client as hypervisor, and each client vm has 40
rbd image, please specify here.
######### test on radosgw with cosbench ##########
cosbench_version=v0.4.2.c2
cosbench_controller=client01
cosbench_driver=client01,client02,client03,client04
cosbench_folder=/opt/cosbench
cosbench_config_dir=/opt/cosbench_config
cosbench_cluster_ip=192.168.5.5
cosbench_admin_ip=192.168.5.1
cosbench_network=192.168.5.0/24
cosbench_auth_username=cosbench:operator
cosbench_auth_password=intel2012
cosbench_controller_proxy=""
cosbench_workers=0,160
cosbench_test_size=128KB
cosbench_rw=write:100
cosbench_containers=r(1,100)
cosbench_objects=r(1,100)
#=====================CEPH CONFIGURATION=====================
[ceph_conf]
public_network = 172.16.96.0/24   #tell CeTune you data network ip,
cluster_network = 172.16.96.0/24  # so CeTune will deploy ceph osd data_network to this nic
```

5.2.2 conf/cases.conf, conf/fio.conf

```
[CeTune_controller]
Cd benchmarking; python run_cases.py --option gen_case
After run gen-case, cases.conf and fio.conf will be generated. You can do modification to these
two files if you necessary.

cat conf/cases.conf
 qemurbd    80     40g      seqwrite        64k         64        0       100  /dev/vdb
 qemurbd    80     40g      randwrite        4k          4        0       100  /dev/vdb
  fiorbd    80     40g      seqwrite        64k         64        0       100    fiorbd
  fiorbd    80     40g      randwrite        4k          4        0       100    fiorbd
cosbench     0     r(1,100)   write:100   128KB      r(1,100)     0       100  cosbench
cosbench   160     r(1,100)   write:100   128KB      r(1,100)     0       100  cosbench
... ...
Cat conf/fio.conf
[global]
```

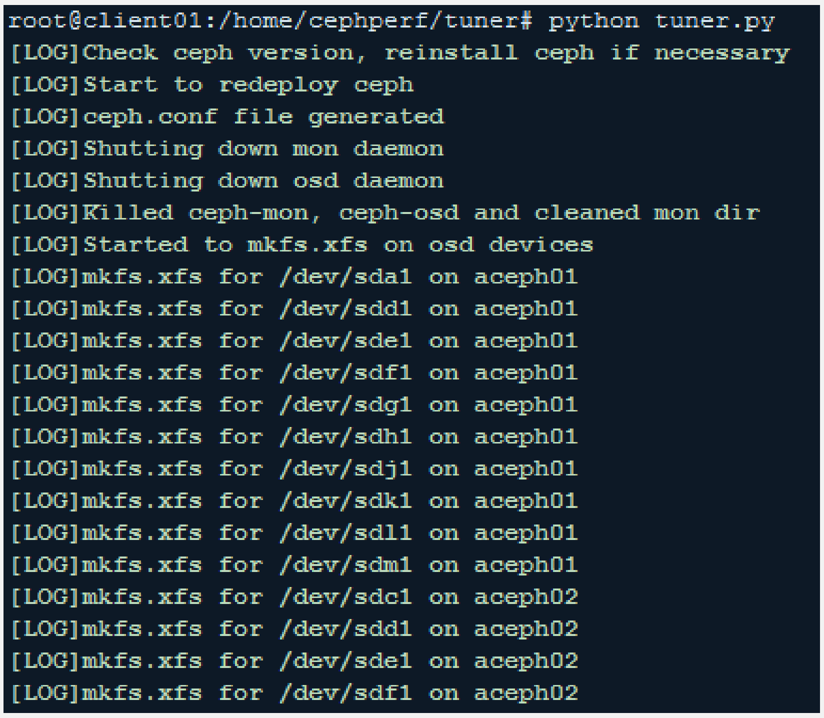
```
  direct=1
  time_based
[seqwrite-64k-qd32-10g-100-300-dev-vdb]
  rw=write
  bs=64k
  iodepth=32
  ramp_time=100
  runtime=300
  iodepth_batch_submit=8
  iodepth_batch_complete=8
  rate=60m
  size=10g
  ioengine=libaio
  filename=/dev/vdb
... ...
```

5.2.3 conf/tuner.yaml

```
[CeTune_controller]
Tuner.yaml is to control CeTune on execution and tuning works
cat conf/tuner.yaml
testjob1:
  workstages: ["deploy", "benchmark"]  #deploy ceph cluster then start benchmark
                                       #you can also only do benchmark, remove deploy
                                       #or only do deploy, remove benchmark.
  pool:                                 # Before benchmark, CeTune will
    rbd:                                # check all the tuning and pool configuration
      size: 2
  disk:
    read_ahead_kb: 2048
  global:
    debug_lockdep: 0/0
    debug_context: 0/0
    debug_crush: 0/0
    debug_buffer: 0/0
    debug_timer: 0/0
    debug_filer: 0/0
    debug_objecter: 0/0
    debug_rados: 0/0
    debug_rbd: 0/0
    debug_journaler: 0/0
    debug_client: 0/0
    debug_osd: 0/0
    debug_optracker: 0/0
    debug_objclass: 0/0
    debug_filestore: 0/0
    debug_journal: 0/0
    debug_ms: 0/0
    debug_monc: 0/0
    debug_tp: 0/0
    debug_auth: 0/0
    debug_finisher: 0/0
    debug_heartbeatmap: 0/0
    debug_perfcounter: 0/0
    debug_asok: 0/0
    debug_throttle: 0/0
    debug_mon: 0/0
    debug_paxos: 0/0
    debug_rgw: 0/0
```

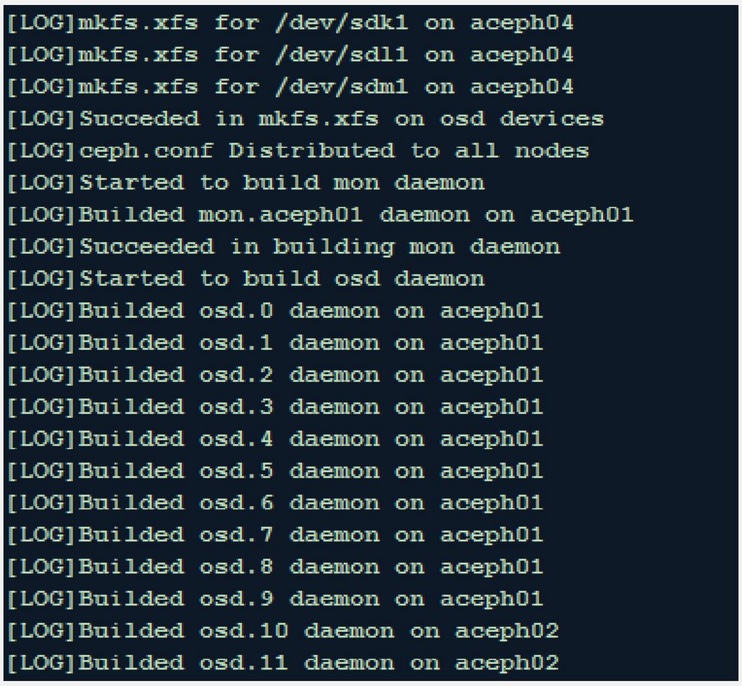
5.3 Run CeTune

```
[CeTune_controller]
Now everything is set, simply kickoff CeTune
cd tuner; python tuner.py
[example]
```



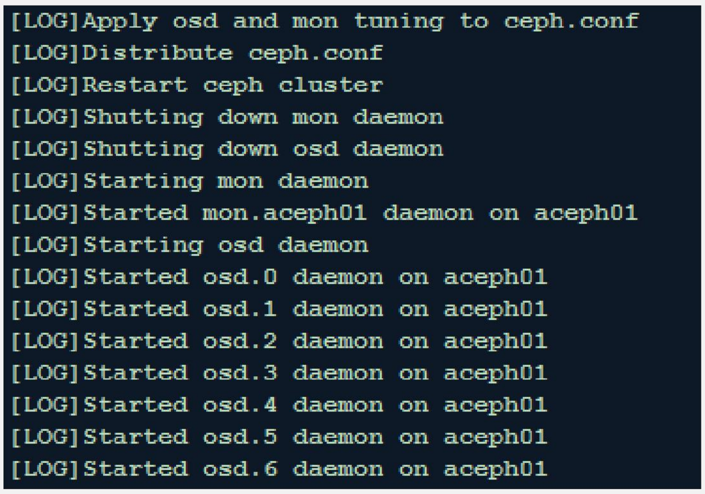
```
root@client01:/home/cephperf/tuner# python tuner.py
[LOG]Check ceph version, reinstall ceph if necessary
[LOG]Start to redeploy ceph
[LOG]ceph.conf file generated
[LOG]Shutting down mon daemon
[LOG]Shutting down osd daemon
[LOG]Killed ceph-mon, ceph-osd and cleaned mon dir
[LOG]Started to mkfs.xfs on osd devices
[LOG]mkfs.xfs for /dev/sda1 on aceph01
[LOG]mkfs.xfs for /dev/sdd1 on aceph01
[LOG]mkfs.xfs for /dev/sde1 on aceph01
[LOG]mkfs.xfs for /dev/sdf1 on aceph01
[LOG]mkfs.xfs for /dev/sdg1 on aceph01
[LOG]mkfs.xfs for /dev/sdh1 on aceph01
[LOG]mkfs.xfs for /dev/sdj1 on aceph01
[LOG]mkfs.xfs for /dev/sdk1 on aceph01
[LOG]mkfs.xfs for /dev/sdl1 on aceph01
[LOG]mkfs.xfs for /dev/sdm1 on aceph01
[LOG]mkfs.xfs for /dev/sdc1 on aceph02
[LOG]mkfs.xfs for /dev/sdd1 on aceph02
[LOG]mkfs.xfs for /dev/sde1 on aceph02
[LOG]mkfs.xfs for /dev/sdf1 on aceph02
```

```
... mkfs.xfs for each osd device
```



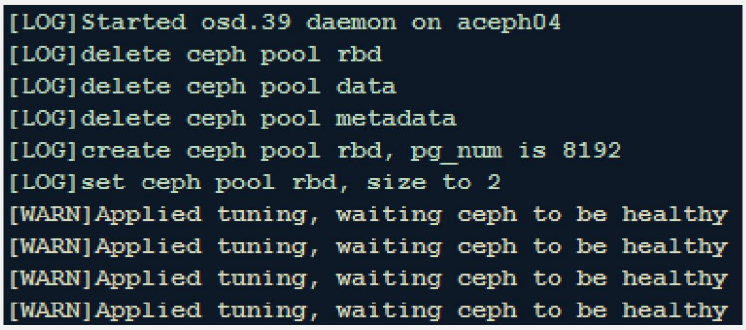
```
[LOG]mkfs.xfs for /dev/sdk1 on aceph04
[LOG]mkfs.xfs for /dev/sdl1 on aceph04
[LOG]mkfs.xfs for /dev/sdm1 on aceph04
[LOG]Succeded in mkfs.xfs on osd devices
[LOG]ceph.conf Distributed to all nodes
[LOG]Started to build mon daemon
[LOG]Builded mon.aceph01 daemon on aceph01
[LOG]Succeeded in building mon daemon
[LOG]Started to build osd daemon
[LOG]Builded osd.0 daemon on aceph01
[LOG]Builded osd.1 daemon on aceph01
[LOG]Builded osd.2 daemon on aceph01
[LOG]Builded osd.3 daemon on aceph01
[LOG]Builded osd.4 daemon on aceph01
[LOG]Builded osd.5 daemon on aceph01
[LOG]Builded osd.6 daemon on aceph01
[LOG]Builded osd.7 daemon on aceph01
[LOG]Builded osd.8 daemon on aceph01
[LOG]Builded osd.9 daemon on aceph01
[LOG]Builded osd.10 daemon on aceph02
[LOG]Builded osd.11 daemon on aceph02
```

... build mon,osd deamon



```
[LOG]Apply osd and mon tuning to ceph.conf
[LOG]Distribute ceph.conf
[LOG]Restart ceph cluster
[LOG]Shutting down mon daemon
[LOG]Shutting down osd daemon
[LOG]Starting mon daemon
[LOG]Started mon.aceph01 daemon on aceph01
[LOG]Starting osd daemon
[LOG]Started osd.0 daemon on aceph01
[LOG]Started osd.1 daemon on aceph01
[LOG]Started osd.2 daemon on aceph01
[LOG]Started osd.3 daemon on aceph01
[LOG]Started osd.4 daemon on aceph01
[LOG]Started osd.5 daemon on aceph01
[LOG]Started osd.6 daemon on aceph01
```

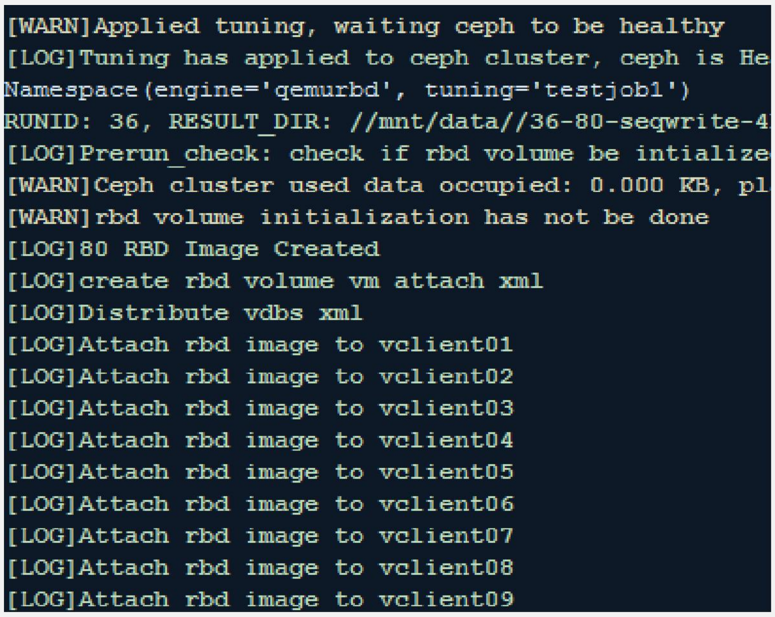
...



```
[LOG]Started osd.39 daemon on aceph04
[LOG]delete ceph pool rbd
[LOG]delete ceph pool data
[LOG]delete ceph pool metadata
[LOG]create ceph pool rbd, pg_num is 8192
[LOG]set ceph pool rbd, size to 2
[WARN]Applied tuning, waiting ceph to be healthy
[WARN]Applied tuning, waiting ceph to be healthy
[WARN]Applied tuning, waiting ceph to be healthy
[WARN]Applied tuning, waiting ceph to be healthy
```

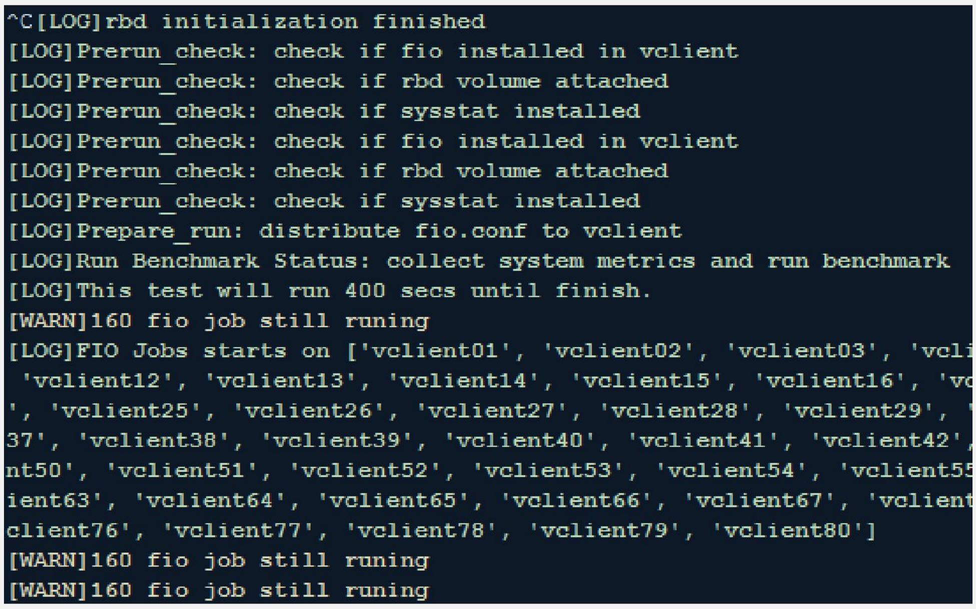
... after Ceph Health being OK, CeTune start to benchmark Ceph

For the first time benchmark, rbd volume hasn’t been created, *CeTune will create rbd and do initialization firstly*



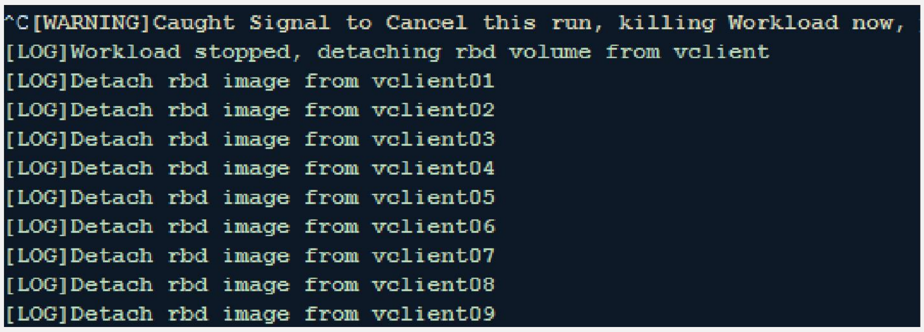
```
[WARN]Applied tuning, waiting ceph to be healthy
[LOG]Tuning has applied to ceph cluster, ceph is He
Namespace(engine='qemurbd', tuning='testjob1')
RUNID: 36, RESULT_DIR: //mnt/data//36-80-seqwrite-4
[LOG]Prerun_check: check if rbd volume be intialize
[WARN]Ceph cluster used data occupied: 0.000 KB, pl
[WARN]rbd volume initialization has not be done
[LOG]80 RBD Image Created
[LOG]create rbd volume vm attach xml
[LOG]Distribute vdbs xml
[LOG]Attach rbd image to vclient01
[LOG]Attach rbd image to vclient02
[LOG]Attach rbd image to vclient03
[LOG]Attach rbd image to vclient04
[LOG]Attach rbd image to vclient05
[LOG]Attach rbd image to vclient06
[LOG]Attach rbd image to vclient07
[LOG]Attach rbd image to vclient08
[LOG]Attach rbd image to vclient09
```

After rbd initialization, start run benchmark.



```
^C[LOG]rbd initialization finished
[LOG]Prerun_check: check if fio installed in vclient
[LOG]Prerun_check: check if rbd volume attached
[LOG]Prerun_check: check if sysstat installed
[LOG]Prerun_check: check if fio installed in vclient
[LOG]Prerun_check: check if rbd volume attached
[LOG]Prerun_check: check if sysstat installed
[LOG]Prepare_run: distribute fio.conf to vclient
[LOG]Run Benchmark Status: collect system metrics and run benchmark
[LOG]This test will run 400 secs until finish.
[WARN]160 fio job still runing
[LOG]FIO Jobs starts on ['vclient01', 'vclient02', 'vclient03', 'vcli
 'vclient12', 'vclient13', 'vclient14', 'vclient15', 'vclient16', 'vc
', 'vclient25', 'vclient26', 'vclient27', 'vclient28', 'vclient29', '
37', 'vclient38', 'vclient39', 'vclient40', 'vclient41', 'vclient42',
nt50', 'vclient51', 'vclient52', 'vclient53', 'vclient54', 'vclient55
ient63', 'vclient64', 'vclient65', 'vclient66', 'vclient67', 'vclient
client76', 'vclient77', 'vclient78', 'vclient79', 'vclient80']
[WARN]160 fio job still runing
[WARN]160 fio job still runing
```

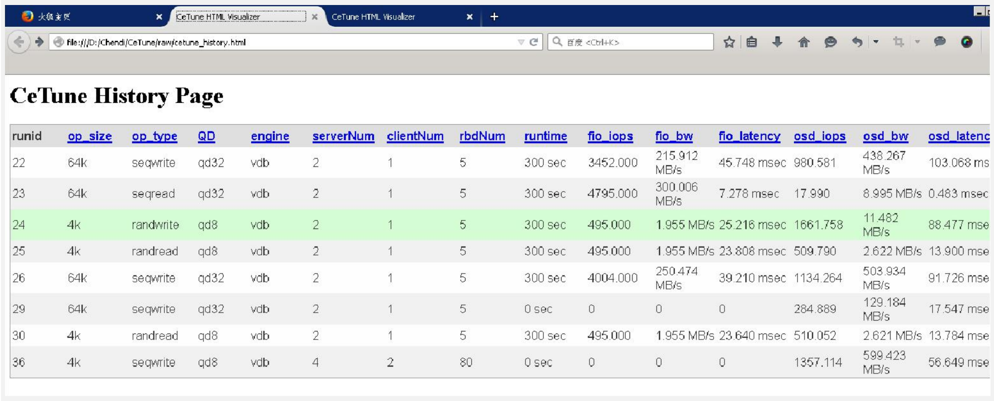
Use ctrl+c to interrupt running test, CeTune will killall workload and detach rbd volume



```
^C[WARNING]Caught Signal to Cancel this run, killing Workload now,
[LOG]Workload stopped, detaching rbd volume from vclient
[LOG]Detach rbd image from vclient01
[LOG]Detach rbd image from vclient02
[LOG]Detach rbd image from vclient03
[LOG]Detach rbd image from vclient04
[LOG]Detach rbd image from vclient05
[LOG]Detach rbd image from vclient06
[LOG]Detach rbd image from vclient07
[LOG]Detach rbd image from vclient08
[LOG]Detach rbd image from vclient09
```

5.4 Result Visualizer

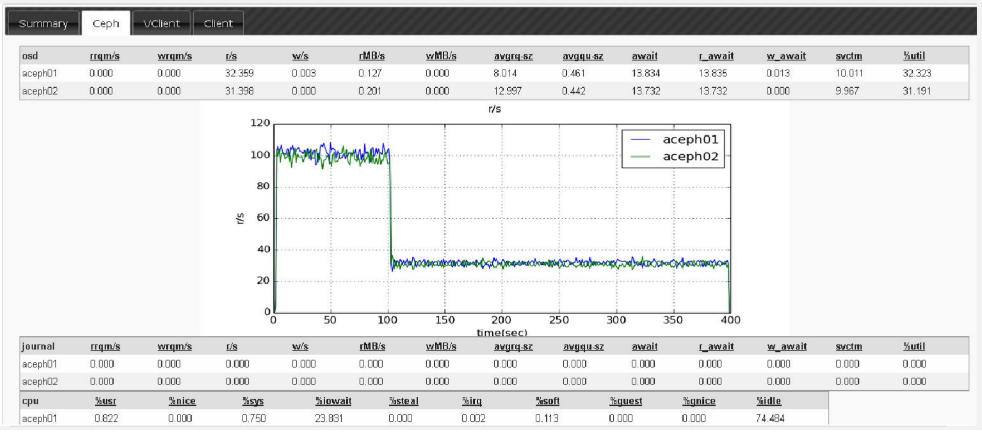
```
Open cetune_history.html at remote_bak_serve
```



CeTune History Page

runid	op_size	op_type	QD	engine	serverNum	clientNum	rbdNum	runtime	fio_iops	fio_bw	fio_latency	osd_iops	osd_bw	osd_latenc
22	64k	seqwrite	qd32	vdb	2	1	5	300 sec	3452.000	215.912 MB/s	45.748 msec	980.581	438.267 MB/s	103.068 ms
23	64k	seqread	qd32	vdb	2	1	5	300 sec	4795.000	300.006 MB/s	7.278 msec	17.990	8.995 MB/s	0.483 msec
24	4k	randwrite	qd8	vdb	2	1	5	300 sec	495.000	1.955 MB/s	25.216 msec	1661.758	11.482 MB/s	88.477 mse
25	4k	randread	qd8	vdb	2	1	5	300 sec	495.000	1.955 MB/s	23.808 msec	509.790	2.622 MB/s	13.900 mse
26	64k	seqwrite	qd32	vdb	2	1	5	300 sec	4004.000	250.474 MB/s	39.210 msec	1134.264	503.934 MB/s	91.726 mse
29	64k	seqwrite	qd32	vdb	2	1	5	0 sec	0	0	0	284.889	129.184 MB/s	17.547 mse
30	4k	randread	qd8	vdb	2	1	5	300 sec	495.000	1.955 MB/s	23.640 msec	510.052	2.621 MB/s	13.784 mse
36	4k	seqwrite	qd8	vdb	4	2	80	0 sec	0	0	0	1357.114	599.423 MB/s	56.649 mse

```
Click one run result,  you can check ceph, vclient and client detailed profiling data there.
```



osd	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
aceph01	0.000	0.000	32.359	0.003	0.127	0.000	8.014	0.461	13.834	13.835	0.013	10.011	32.323
aceph02	0.000	0.000	31.398	0.000	0.201	0.000	12.997	0.442	13.732	13.732	0.000	9.967	31.191

journal	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
aceph01	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
aceph02	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

cpu	%usr	%nice	%sys	%iowait	%steal	%irq	%soft	%guest	%gnice	%idle
aceph01	0.822	0.000	0.750	23.831	0.000	0.002	0.113	0.000	0.000	74.484

5.5 Kickoff each module

Cetune modules like ‘deploy’, ‘benchmark’, ‘analyze’, ‘visualize’ can also run separately. Using below command to run

```
# Install ceph package
cd deploy; python run_deploy.py install_binary --version ${version}
# Uninstall ceph package
cd deploy; python run_deploy.py uninstall_binary
# Redeploy ceph cluster
cd deploy; python run_deploy.py redeploy
# Restart ceph cluster
cd deploy; python run_deploy.py restart
# Deploy ceph radosgw
cd deploy; python run_deploy.py deploy_rgw
# Generate ceph conf from tuner.yaml
cd deploy; python run_deploy.py gen_cephconf
# Send generated ceph conf to all osd and client node.
cd deploy; python run_deploy.py distribute_conf
```

```
# Generate test cases for cosbench and fio
cd benchmarking; python run_cases.py -option gen_case
# Run benchmark following the sequence in ../conf/cases.conf
cd benchmarking; python run_cases.py
# Do analyze on one cetune test result
cd analyzer; python analyzer.py -path ${path} process_data
# Do visualize on one cetune analyzed result, need result.json doc under ${path}
cd visualizer; python visualizer.py -path {$path} generate_summary_page
```

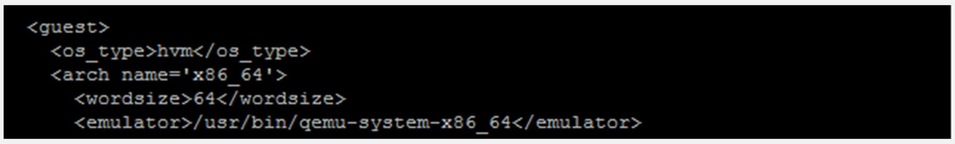
6. Appendix

In the processing of installing ceph you may encounter the follow problems, and here are some tips for solving them.

6.1 OS type 'hvm' unknow

When you creating vclient by virsh and encounter the error below:
error: unknown OS type hvm
error: internal error: no supported architecture for os type 'hvm'

```
[solution]
#Check virsh capabilities fisrt
virsh capabilities | grep os_type
<os_type>hvm</os_type>
#if not get above data, which means kvm is not installed correctly

apt-get/yum install qemu_kvm
modprobe kvm
modprobe kvm_intel
lsmod | grep kvm
kvm-ok
virsh capabilities  #then you can find os type is hvm
```



```
<guest>
  <os_type>hvm</os_type>
  <arch name='x86_64'>
    <wordsize>64</wordsize>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
```

6.2 operation failed: open disk image file failed

```
#Firewall may not being give privilege to virsh
/etc/init.d/apparmor teardown or aa-complain /etc/init.d/libvirt-bin
/etc/init.d/libvirt-bin restart
Re-create vclient
```

6.3 How to install fio with rbd engine

```
#Make sure librbd-dev and librados-dev is installed, or you won't be able to configure fio
```

```
with rbdengine
dpkg -l | grep librbd-dev
ii  librbd-dev   0.80.9-1trusty  amd64  RADOS block device client library (development files)
dpkg -l | grep librados-dev
ii  librados-dev 0.80.9-1trusty  amd64  RADOS distributed object store client library
(development files)
git clone https://github.com/axboe/fio.git
cd fio
./configure
make
make install
fio --enghelp
Available IO engines:
     cpuio
     mmap
     sync
     psync
     vsync
     pvsync
     null
     net
     netsplice
     libaio
     posixaio
     falloc
     e4defrag
     splice
     rbd
     sg
     binject
```

6.4 pgs stuck unclean

```
   root@aceph01:/var/log/ceph# ceph -s
    cluster 2f96f09f-d911-4c69-8236-053e6d15fb11
     health HEALTH_WARN 1024 pgs stuck unclean
     monmap e1: 1 mons at {aceph01=172.16.96.11:6789/0}, election epoch 2, quorum 0 aceph01
     osdmap e70: 16 osds: 16 up, 16 in
      pgmap v136: 1024 pgs, 1 pools, 0 bytes data, 0 objects
            587 MB used, 14896 GB / 14896 GB avail
                1024 active
```

```
[solution]
```

```
ceph osd dump | grep pool    #dump the information of pool
ceph osd tree   #check osd status
#if all above is ok, then
ceph osd pool set rbd size 1
#wait ceph to be healthy, then
ceph osd pool set rbd size ${replica_size}
```