**Created a sample erasure code :**

The simplest erasure coded pool is equivalent to [RAID5](https://en.wikipedia.org/wiki/Standard_RAID_levels#RAID_5) and requires at least three hosts:

We are created and modified the osds as five hosts in osd tree, Modified osds are given below :

[cephuser@ceph-admin cluster]$ ceph osd tree

ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF

-1 0.51195 root default

-3 0.51195 rack rack1

-5 0.03799 host row1

0 hdd 0.01900 osd.0 up 1.00000 1.00000

1 hdd 0.01900 osd.1 up 1.00000 1.00000

-6 0.11499 host row2

2 hdd 0.01700 osd.2 up 1.00000 1.00000

5 hdd 0.09799 osd.5 up 1.00000 1.00000

-9 0.31699 host row3

7 hdd 0.26099 osd.7 up 1.00000 1.00000

8 hdd 0.05600 osd.8 up 1.00000 1.00000

-11 0.03099 host row4

3 hdd 0.00999 osd.3 up 1.00000 1.00000

4 hdd 0.02100 osd.4 up 1.00000 1.00000

-12 0.01099 host row5

6 hdd 0.01099 osd.6 up 1.00000 1.00000

These hosts are normally created by crush command of buckets and moved to rack section

Now creating a new profile of erasure code and setting the value of the EC , Sample output given below:

ceph osd erasure-code-profile set evolution k=3 m=2 crush-failure-domain=host

Note:The EC to creates the own crush\_rule ,based on the value

Here evolution is our profile name and the most important parameters of the profile are K, M and crush-failure-domain because they define the storage overhead and the data durability.

The OBJECT-A object will be divided in two (K=2) and one additional chunks will be created (M=1). The value of M defines how many OSD can be lost simultaneously without losing any data. The crush-failure-domain=host will create a CRUSH rule that ensures no two chunks are stored in the same rack.

**Test Case:**

**Now creating the pool with erasure code :**

ceph osd pool create ecpool 128 128 erasure evolution

**created pool to be checked :**

[cephuser@ceph-admin cluster]$ ceph osd lspools

82 metadata,94 row1,96 .rgw.root,97 default.rgw.control,98 default.rgw.meta,99 default.rgw.log,100 default.rgw.buckets.index,101 default.rgw.buckets.data,114 ecpool

**Now we put the object in ecpool:**

rados -p ecpool put object-A object-A

**Object is successfully placed on pools with erasure code :**

[cephuser@ceph-admin cluster]$ rados -p ecpool ls

object-A

**Check on pool detail :**

[cephuser@ceph-admin cluster]$ ceph osd dump | grep ecpool

pool 114 'ecpool' erasure size 5 min\_size 3 crush\_rule 4 object\_hash rjenkins pg\_num 128 pgp\_num 128 last\_change 1763 flags hashpspool,ec\_overwrites stripe\_width 8192

Output of the object placed shown below

OBJECT-A

name

ABCDEFGHI

content

row5

row4

row3

row2

row1

UVW  
(4)

GHI  
(3)

Encode (2,1)

XYZ  
(5)

ABC  
(1)

DEF  
 (2)

If down the third osd, data will gone the one of the part of data that means one chunk is gone

OBJECT-A

name

ABCDEFGHI

content

row5

row4

row3

row2

row1

UVW  
(4)

GHI  
(3)

Encode (2,1)

XYZ  
(5)

ABC  
(1)

DEF  
 (2)

Here the data is dynamically reconstructed based on partially content of the chunk , its based on reed\_sol\_mann technique

OBJECT-A

name

ABCDEFGHI

content

row5

row4

row3

row2

row1

UVW  
(4)

GHI  
(3)

Encode (2,1)

GHI

XYZ

1

1

XYZ  
(5)

ABC  
(1)

DEF  
 (2)

Now We can check where the data is stored:

[cephuser@ceph-admin cluster]$ ceph osd map ecpool object-A

osdmap e1835 pool 'ecpool' (117) object 'object-A' -> pg 117.b301e3e8 (117.8) -> up ([7,6,5,3,1], p7) acting ([7,6,5,3,1], p7)

From the above output these are the 5 OSD on which the file “object-A” has been

spread = [7,6,5,3,1]

Intentionally Breaking **OSD to test EC reliability:**

**Breaking the 1st  OSD**

[root@osd1 ~]# systemctl stop ceph-osd@1

[root@osd1 ~]# systemctl status ceph-osd@1

● ceph-osd@1.service - Ceph object storage daemon osd.1

Loaded: loaded (/usr/lib/systemd/system/ceph-osd@.service; enabled; vendor preset: disabled)

Active: inactive (dead) since Thu 2018-05-10 01:24:24 EDT; 19s ago

Process: 1404 ExecStart=/usr/bin/ceph-osd -f --cluster ${CLUSTER} --id %i --setuser ceph --setgroup ceph (code=exited, status=0/SUCCESS)

Process: 1400 ExecStartPre=/usr/lib/ceph/ceph-osd-prestart.sh --cluster ${CLUSTER} --id %i (code=exited, status=0/SUCCESS)

Main PID: 1404 (code=exited, status=0/SUCCESS)

**Breaking the 3rd OSD**

[root@osd2 ~]# systemctl stop ceph-osd@3

[root@osd2 ~]# systemctl status ceph-osd@3

ceph-osd@3.service - Ceph object storage daemon osd.3

Loaded: loaded (/usr/lib/systemd/system/ceph-osd@.service; disabled; vendor preset: disabled)

Active: inactive (dead)

May 10 00:54:37 osd2 systemd[1]: Starting Ceph object storage daemon osd.3...  
May 10 00:54:37 osd2 systemd[1]: Started Ceph object storage daemon osd.3.  
May 10 00:54:38 osd2 ceph-osd[1088]: starting osd.3 at - osd\_data /var/lib/ceph/osd/ceph-3 /var/lib/ceph/osd/ceph-3/journal  
May 10 00:55:00 osd2 ceph-osd[1088]: 2018-05-10 00:55:00.823931 7f5176e59d00 -1 osd.3 1766 log\_to\_monitors {default=true}  
May 10 01:52:46 osd2 systemd[1]: Stopping Ceph object storage daemon osd.3..

**Checking the object placement:**

[cephuser@ceph-admin cluster]$ ceph osd map ecpool object-A

osdmap e1858 pool 'ecpool' (117) object 'object-A' -> pg 117.b301e3e8 (117.8) -> up ([7,6,5,4,0], p7) acting ([7,6,5,4,0], p7)

**Still the osd are down :**

[cephuser@ceph-admin cluster]$ ceph osd tree

ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF

-1 0.51199 root default

-3 0.51199 rack rack1

-5 0.03799 host row1

0 hdd 0.01900 osd.0 up 1.00000 1.00000

1 hdd 0.01900 osd.1 down 0 1.00000

-6 0.11499 host row2

2 hdd 0.01700 osd.2 up 1.00000 1.00000

5 hdd 0.09799 osd.5 up 1.00000 1.00000

-9 0.31699 host row3

7 hdd 0.26099 osd.7 up 1.00000 1.00000

8 hdd 0.05600 osd.8 up 1.00000 1.00000

-11 0.03099 host row4

3 hdd 0.00999 osd.3 down 0 1.00000

4 hdd 0.02100 osd.4 up 1.00000 1.00000

-12 0.01099 host row5

6 hdd 0.01099 osd.6 up 1.00000 1.00000

**Once up it will back on the lost osds :**

[cephuser@ceph-admin cluster]$ ceph osd map ecpool object-A

osdmap e1868 pool 'ecpool' (117) object 'object-A' -> pg 117.b301e3e8 (117.8) -> up ([7,6,5,3,1], p7) acting ([7,6,5,3,1], p7)

**Erasure codings are intelligent , it know when you lose data or coding chunks of a file. As soon as chunks are lost , it immediately create exactly same on to new OSD. In this example you can see , OSD.1 and OSD.3 went down , so ceph intelligently recovers the failed chunk on to OSD.4 and OSD.0 ( which is a new OSD )**

**Creating Sample Replicated rule:**

[cephuser@ceph-admin cluster]$ ceph osd tree

ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF

-1 0.51199 root default

-3 0.51199 rack rack1

-5 0.03799 host row1

0 hdd 0.01900 osd.0 up 1.00000 1.00000

1 hdd 0.01900 osd.1 up 1.00000 1.00000

-6 0.11499 host row2

2 hdd 0.01700 osd.2 up 1.00000 1.00000

5 hdd 0.09799 osd.5 up 1.00000 1.00000

-9 0.31699 host row3

7 hdd 0.26099 osd.7 up 1.00000 1.00000

8 hdd 0.05600 osd.8 up 1.00000 1.00000

-11 0.03099 host row4

3 hdd 0.00999 osd.3 up 1.00000 1.00000

4 hdd 0.02100 osd.4 up 1.00000 1.00000

-12 0.01099 host row5

6 hdd 0.01099 osd.6 up 1.00000 1.00000

Above tree we data taken all hosts , it means data spread all osds (based on default rule)

**Now we are create a pool for first row with 256 pgs**

ceph osd pool create row1 256

**Next we create a new crush ruleset(manually)**

ceph osd getcrushmap –o newcompile1152018

**Decompile the compiled file**

crushtool –d newcompile1152018 –o newdecompile1152018

**Then edit the decompile file to add a new rule below format**

rule row1 {

id 1

type replicated

min\_size 1

max\_size 10

step take row1

step choose firstn 0 type osd

step emit

}

It means the replicated method are from 1 to 10 between values to be taken on row1 , that row1 under all osds are takened

**Again modified decompile to change compiled file**

crushtool –c newdecompile1152018 –o newcompile

**Then we are import the newcompile file to crushmap**

ceph osd setcrushmap –i newcompile

**Now check the rule on map changes:**

ceph osd crush rule dump row1

[cephuser@ceph-admin cluster]$ ceph osd crush rule dump row1

{

"rule\_id": 1,

"rule\_name": "row1",

"ruleset": 1,

"type": 1,

"min\_size": 1,

"max\_size": 10,

"steps": [

{

"op": "take",

"item": -5,

"item\_name": "row1"

},

{

"op": "choose\_firstn",

"num": 0,

"type": "osd"

},

{

"op": "emit"

}

]

}

**Test Case:**

**Next we are set the rule on created pool row1**

ceph osd pool set row1 crush\_rule row1

**Then we are creating the file sysem to client side with MDS**

ceph fs new cephfs metadata row1

**Now we are mounted the filesystem on client side**

sudo mkdir -p /mnt/fuse

sudo ceph-fuse -m mon1:6789 /mnt/fuse

**Here put the data on fuse mountpoint**

scp -r /ftp/tvn/SourceBackup/CHIPS [root@192.168.1.166:/mnt/fuse](mailto:root@192.168.1.166:/mnt/fuse)

**Checking the data placement after completion :**

[cephuser@ceph-admin cluster]$ sh metadata.sh

Enter Pool name :row1

0,1

0,1

0,1

0,1

0,1

0,1

0,1

1,0

1,0

1,0

0,1

0,1

Now the data is palced based on crush rule to make only on row1 osds only .