PLACEMENT GROUPS

A PRESELECTION OF PG_NUM

When creating a new pool with:

```
ceph osd pool create {pool-name} pg_num
```

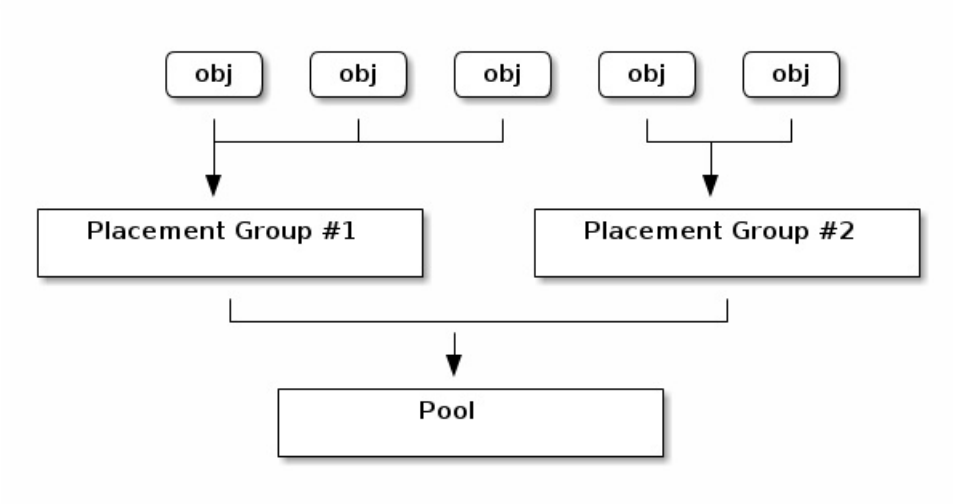
it is mandatory to choose the value of `pg_num` because it cannot be calculated automatically. Here are a few values commonly used:

- Less than 5 OSDs set `pg_num` to 128
- Between 5 and 10 OSDs set `pg_num` to 512
- Between 10 and 50 OSDs set `pg_num` to 1024
- If you have more than 50 OSDs, you need to understand the tradeoffs and how to calculate the `pg_num` value by yourself
- For calculating `pg_num` value by yourself please take help of pgcalc tool

As the number of OSDs increases, chosing the right value for pg_num becomes more important because it has a significant influence on the behavior of the cluster as well as the durability of the data when something goes wrong (i.e. the probability that a catastrophic event leads to data loss).

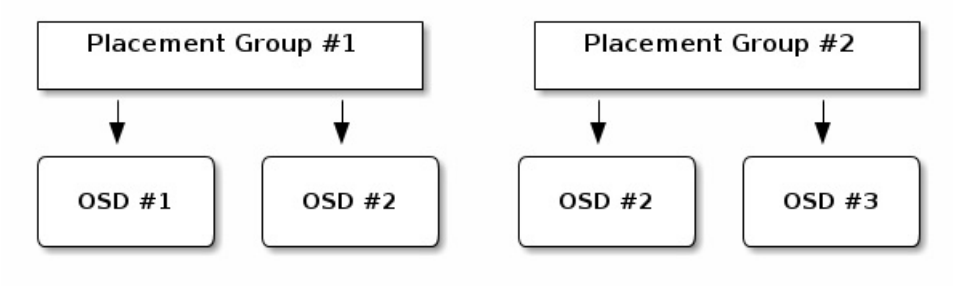
HOW ARE PLACEMENT GROUPS USED ?

A placement group (PG) aggregates objects within a pool because tracking object placement and object metadata on a per-object basis is computationally expensive–i.e., a system with millions of objects cannot realistically track placement on a per-object basis.



The Ceph client will calculate which placement group an object should be in. It does this by hashing the object ID and applying an operation based on the number of PGs in the defined pool and the ID of the pool. See Mapping PGs to OSDs for details.

The object's contents within a placement group are stored in a set of OSDs. For instance, in a replicated pool of size two, each placement group will store objects on two OSDs, as shown below.



Should OSD #2 fail, another will be assigned to Placement Group #1 and will be filled with copies of all objects in OSD #1. If

the pool size is changed from two to three, an additional OSD will be assigned to the placement group and will receive copies of all objects in the placement group.

Placement groups do not own the OSD, they share it with other placement groups from the same pool or even other pools. If OSD #2 fails, the Placement Group #2 will also have to restore copies of objects, using OSD #3.

When the number of placement groups increases, the new placement groups will be assigned OSDs. The result of the CRUSH function will also change and some objects from the former placement groups will be copied over to the new Placement Groups and removed from the old ones.

PLACEMENT GROUPS TRADEOFFS

Data durability and even distribution among all OSDs call for more placement groups but their number should be reduced to the minimum to save CPU and memory.

DATA DURABILITY

After an OSD fails, the risk of data loss increases until the data it contained is fully recovered. Let's imagine a scenario that causes permanent data loss in a single placement group:

- The OSD fails and all copies of the object it contains are lost. For all objects within the placement group the number of replica suddenly drops from three to two.
- Ceph starts recovery for this placement group by choosing a new OSD to re-create the third copy of all objects.
- Another OSD, within the same placement group, fails before the new OSD is fully populated with the third copy. Some objects will then only have one surviving copies.
- Ceph picks yet another OSD and keeps copying objects to restore the desired number of copies.
- A third OSD, within the same placement group, fails before recovery is complete. If this OSD contained the only remaining copy of an object, it is permanently lost.

In a cluster containing 10 OSDs with 512 placement groups in a three replica pool, CRUSH will give each placement groups three OSDs. In the end, each OSDs will end up hosting (512 * 3) / 10 = ~150 Placement Groups. When the first OSD fails, the above scenario will therefore start recovery for all 150 placement groups at the same time.

The 150 placement groups being recovered are likely to be homogeneously spread over the 9 remaining OSDs. Each remaining OSD is therefore likely to send copies of objects to all others and also receive some new objects to be stored because they became part of a new placement group.

The amount of time it takes for this recovery to complete entirely depends on the architecture of the Ceph cluster. Let say each OSD is hosted by a 1TB SSD on a single machine and all of them are connected to a 10Gb/s switch and the recovery for a single OSD completes within M minutes. If there are two OSDs per machine using spinners with no SSD journal and a 1Gb/s switch, it will at least be an order of magnitude slower.

In a cluster of this size, the number of placement groups has almost no influence on data durability. It could be 128 or 8192 and the recovery would not be slower or faster.

However, growing the same Ceph cluster to 20 OSDs instead of 10 OSDs is likely to speed up recovery and therefore improve data durability significantly. Each OSD now participates in only ~75 placement groups instead of ~150 when there were only 10 OSDs and it will still require all 19 remaining OSDs to perform the same amount of object copies in order to recover. But where 10 OSDs had to copy approximately 100GB each, they now have to copy 50GB each instead. If the network was the bottleneck, recovery will happen twice as fast. In other words, recovery goes faster when the number of OSDs increases.

If this cluster grows to 40 OSDs, each of them will only host ~35 placement groups. If an OSD dies, recovery will keep going faster unless it is blocked by another bottleneck. However, if this cluster grows to 200 OSDs, each of them will only host ~7 placement groups. If an OSD dies, recovery will happen between at most of ~21 (7 * 3) OSDs in these placement groups: recovery will take longer than when there were 40 OSDs, meaning the number of placement groups should be increased.

No matter how short the recovery time is, there is a chance for a second OSD to fail while it is in progress. In the 10 OSDs cluster described above, if any of them fail, then ~17 placement groups (i.e. ~150 / 9 placement groups being recovered) will only have one surviving copy. And if any of the 8 remaining OSD fail, the last objects of two placement groups are likely to be lost (i.e. ~17 / 8 placement groups with only one remaining copy being recovered).

When the size of the cluster grows to 20 OSDs, the number of Placement Groups damaged by the loss of three OSDs drops. The second OSD lost will degrade ~4 (i.e. ~75 / 19 placement groups being recovered) instead of ~17 and the third OSD lost will only lose data if it is one of the four OSDs containing the surviving copy. In other words, if the probability of losing one OSD is 0.0001% during the recovery time frame, it goes from 17 * 10 * 0.0001% in the cluster with 10 OSDs to 4 * 20 * 0.0001% in the cluster with 20 OSDs.

In a nutshell, more OSDs mean faster recovery and a lower risk of cascading failures leading to the permanent loss of a Placement Group. Having 512 or 4096 Placement Groups is roughly equivalent in a cluster with less than 50 OSDs as far as data durability is concerned.

Note: It may take a long time for a new OSD added to the cluster to be populated with placement groups that were assigned to it. However there is no degradation of any object and it has no impact on the durability of the data contained in the Cluster.

OBJECT DISTRIBUTION WITHIN A POOL

Ideally objects are evenly distributed in each placement group. Since CRUSH computes the placement group for each object, but does not actually know how much data is stored in each OSD within this placement group, the ratio between the number of placement groups and the number of OSDs may influence the distribution of the data significantly.

For instance, if there was a single placement group for ten OSDs in a three replica pool, only three OSD would be used because CRUSH would have no other choice. When more placement groups are available, objects are more likely to be evenly spread among them. CRUSH also makes every effort to evenly spread OSDs among all existing Placement Groups.

As long as there are one or two orders of magnitude more Placement Groups than OSDs, the distribution should be even. For instance, 300 placement groups for 3 OSDs, 1000 placement groups for 10 OSDs etc.

Uneven data distribution can be caused by factors other than the ratio between OSDs and placement groups. Since CRUSH does not take into account the size of the objects, a few very large objects may create an imbalance. Let say one million 4K objects totaling 4GB are evenly spread among 1000 placement groups on 10 OSDs. They will use 4GB / 10 = 400MB on each OSD. If one 400MB object is added to the pool, the three OSDs supporting the placement group in which the object has been placed will be filled with 400MB + 400MB = 800MB while the seven others will remain occupied with only 400MB.

MEMORY, CPU AND NETWORK USAGE

For each placement group, OSDs and MONs need memory, network and CPU at all times and even more during recovery. Sharing this overhead by clustering objects within a placement group is one of the main reasons they exist.

Minimizing the number of placement groups saves significant amounts of resources.

CHOOSING THE NUMBER OF PLACEMENT GROUPS

If you have more than 50 OSDs, we recommend approximately 50-100 placement groups per OSD to balance out resource usage, data durability and distribution. If you have less than 50 OSDs, choosing among the preselection above is best. For a single pool of objects, you can use the following formula to get a baseline:

```
             (OSDs * 100)
Total PGs =  ------------
              pool size
```

Where **pool size** is either the number of replicas for replicated pools or the K+M sum for erasure coded pools (as returned by **ceph osd erasure-code-profile get**).

You should then check if the result makes sense with the way you designed your Ceph cluster to maximize data durability, object distribution and minimize resource usage.

The result should be **rounded up to the nearest power of two.** Rounding up is optional, but recommended for CRUSH to evenly balance the number of objects among placement groups.

As an example, for a cluster with 200 OSDs and a pool size of 3 replicas, you would estimate your number of PGs as follows:

```
(200 * 100)
----------- = 6667. Nearest power of 2: 8192
     3
```

When using multiple data pools for storing objects, you need to ensure that you balance the number of placement groups per pool with the number of placement groups per OSD so that you arrive at a reasonable total number of placement groups that provides reasonably low variance per OSD without taxing system resources or making the peering process too slow.

For instance a cluster of 10 pools each with 512 placement groups on ten OSDs is a total of 5,120 placement groups spread

over ten OSDs, that is 512 placement groups per OSD. That does not use too many resources. However, if 1,000 pools were created with 512 placement groups each, the OSDs will handle ~50,000 placement groups each and it would require significantly more resources and time for peering.

You may find the PGCalc tool helpful.

SET THE NUMBER OF PLACEMENT GROUPS

To set the number of placement groups in a pool, you must specify the number of placement groups at the time you create the pool. See Create a Pool for details. Once you have set placement groups for a pool, you may increase the number of placement groups (but you cannot decrease the number of placement groups). To increase the number of placement groups, execute the following:

```
ceph osd pool set {pool-name} pg_num {pg_num}
```

Once you increase the number of placement groups, you must also increase the number of placement groups for placement (`pgp_num`) before your cluster will rebalance. The `pgp_num` will be the number of placement groups that will be considered for placement by the CRUSH algorithm. Increasing `pg_num` splits the placement groups but data will not be migrated to the newer placement groups until placement groups for placement, ie. `pgp_num` is increased. The `pgp_num` should be equal to the `pg_num`. To increase the number of placement groups for placement, execute the following:

```
ceph osd pool set {pool-name} pgp_num {pgp_num}
```

GET THE NUMBER OF PLACEMENT GROUPS

To get the number of placement groups in a pool, execute the following:

```
ceph osd pool get {pool-name} pg_num
```

GET A CLUSTER'S PG STATISTICS

To get the statistics for the placement groups in your cluster, execute the following:

```
ceph pg dump [--format {format}]
```

Valid formats are `plain` (default) and `json`.

GET STATISTICS FOR STUCK PGS

To get the statistics for all placement groups stuck in a specified state, execute the following:

```
ceph pg dump_stuck inactive|unclean|stale|undersized|degraded [--format <format>] [-t|--thres
```

Inactive Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come up and in.

Unclean Placement groups contain objects that are not replicated the desired number of times. They should be recovering.

Stale Placement groups are in an unknown state - the OSDs that host them have not reported to the monitor cluster in a while (configured by `mon_osd_report_timeout`).

Valid formats are `plain` (default) and `json`. The threshold defines the minimum number of seconds the placement group is stuck before including it in the returned statistics (default 300 seconds).

GET A PG MAP

To get the placement group map for a particular placement group, execute the following:

```
ceph pg map {pg-id}
```

For example:

```
ceph pg map 1.6c
```

Ceph will return the placement group map, the placement group, and the OSD status:

```
osdmap e13 pg 1.6c (1.6c) -> up [1,0] acting [1,0]
```

GET A PGS STATISTICS

To retrieve statistics for a particular placement group, execute the following:

```
ceph pg {pg-id} query
```

SCRUB A PLACEMENT GROUP

To scrub a placement group, execute the following:

```
ceph pg scrub {pg-id}
```

Ceph checks the primary and any replica nodes, generates a catalog of all objects in the placement group and compares them to ensure that no objects are missing or mismatched, and their contents are consistent. Assuming the replicas all match, a final semantic sweep ensures that all of the snapshot-related object metadata is consistent. Errors are reported via logs.

PRIORITIZE BACKFILL/RECOVERY OF A PLACEMENT GROUP(S)

You may run into a situation where a bunch of placement groups will require recovery and/or backfill, and some particular groups hold data more important than others (for example, those PGs may hold data for images used by running machines and other PGs may be used by inactive machines/less relevant data). In that case, you may want to prioritize recovery of those groups so performance and/or availability of data stored on those groups is restored earlier. To do this (mark particular placement group(s) as prioritized during backfill or recovery), execute the following:

```
ceph pg force-recovery {pg-id} [{pg-id #2}] [{pg-id #3} ...]
ceph pg force-backfill {pg-id} [{pg-id #2}] [{pg-id #3} ...]
```

This will cause Ceph to perform recovery or backfill on specified placement groups first, before other placement groups. This does not interrupt currently ongoing backfills or recovery, but causes specified PGs to be processed as soon as possible. If you change your mind or prioritize wrong groups, use:

```
ceph pg cancel-force-recovery {pg-id} [{pg-id #2}] [{pg-id #3} ...]
ceph pg cancel-force-backfill {pg-id} [{pg-id #2}] [{pg-id #3} ...]
```

This will remove “force” flag from those PGs and they will be processed in default order. Again, this doesn’t affect currently processed placement group, only those that are still queued.

The “force” flag is cleared automatically after recovery or backfill of group is done.

REVERT LOST

If the cluster has lost one or more objects, and you have decided to abandon the search for the lost data, you must mark the unfound objects as `lost`.

If all possible locations have been queried and objects are still lost, you may have to give up on the lost objects. This is possible given unusual combinations of failures that allow the cluster to learn about writes that were performed before the writes themselves are recovered.

Currently the only supported option is “revert”, which will either roll back to a previous version of the object or (if it was a new object) forget about it entirely. To mark the “unfound” objects as “lost”, execute the following:

```
ceph pg {pg-id} mark_unfound_lost revert|delete
```

Important: Use this feature with caution, because it may confuse applications that expect the object(s) to exist.