CONFIGURING MULTIPLE ACTIVE MDS DAEMONS

Also known as: multi-mds, active-active MDS

Each CephFS filesystem is configured for a single active MDS daemon by default. To scale metadata performance for large scale systems, you may enable multiple active MDS daemons, which will share the metadata workload with one another.

WHEN SHOULD I USE MULTIPLE ACTIVE MDS DAEMONS?

You should configure multiple active MDS daemons when your metadata performance is bottlenecked on the single MDS that runs by default.

Adding more daemons may not increase performance on all workloads. Typically, a single application running on a single client will not benefit from an increased number of MDS daemons unless the application is doing a lot of metadata operations in parallel.

Workloads that typically benefit from a larger number of active MDS daemons are those with many clients, perhaps working on many separate directories.

INCREASING THE MDS ACTIVE CLUSTER SIZE

Each CephFS filesystem has a *max_mds* setting, which controls how many ranks will be created. The actual number of ranks in the filesystem will only be increased if a spare daemon is available to take on the new rank. For example, if there is only one MDS daemon running, and max_mds is set to two, no second rank will be created.

Before `max_mds` can be increased, the `allow_multimds` flag must be set. The following command sets this flag for a filesystem instance.

```
# ceph fs set <fs_name> allow_multimds true --yes-i-really-mean-it
```

Set `max_mds` to the desired number of ranks. In the following examples the “fsmap” line of “ceph status” is shown to illustrate the expected result of commands.

```
# fsmap e5: 1/1/1 up {0=a=up:active}, 2 up:standby

ceph fs set <fs_name> max_mds 2

# fsmap e8: 2/2/2 up {0=a=up:active,1=c=up:creating}, 1 up:standby
# fsmap e9: 2/2/2 up {0=a=up:active,1=c=up:active}, 1 up:standby
```

The newly created rank (1) will pass through the ‘creating’ state and then enter this ‘active state’.

STANDBY DAEMONS

Even with multiple active MDS daemons, a highly available system **still requires standby daemons** to take over if any of the servers running an active daemon fail.

Consequently, the practical maximum of `max_mds` for highly available systems is one less than the total number of MDS servers in your system.

To remain available in the event of multiple server failures, increase the number of standby daemons in the system to match the number of server failures you wish to withstand.

DECREASING THE NUMBER OF RANKS

All ranks, including the rank(s) to be removed must first be active. This means that you must have at least max_mds MDS daemons available.

First, set max_mds to a lower number, for example we might go back to having just a single active MDS:

```
# fsmap e9: 2/2/2 up {0=a=up:active,1=c=up:active}, 1 up:standby
ceph fs set <fs_name> max_mds 1
# fsmap e10: 2/2/1 up {0=a=up:active,1=c=up:active}, 1 up:standby
```

Note that we still have two active MDSs: the ranks still exist even though we have decreased max_mds, because max_mds only restricts creation of new ranks.

Next, use the `ceph mds deactivate <role>` command to remove the unneeded rank:

```
ceph mds deactivate cephfs_a:1
telling mds.1:1 172.21.9.34:6806/837679928 to deactivate

# fsmap e11: 2/2/1 up {0=a=up:active,1=c=up:stopping}, 1 up:standby
# fsmap e12: 1/1/1 up {0=a=up:active}, 1 up:standby
# fsmap e13: 1/1/1 up {0=a=up:active}, 2 up:standby
```

See CephFS Administrative commands for more details which forms `<role>` can take.

The deactivated rank will first enter the stopping state for a period of time while it hands off its share of the metadata to the remaining active daemons. This phase can take from seconds to minutes. If the MDS appears to be stuck in the stopping state then that should be investigated as a possible bug.

If an MDS daemon crashes or is killed while in the ‘stopping’ state, a standby will take over and the rank will go back to ‘active’. You can try to deactivate it again once it has come back up.

When a daemon finishes stopping, it will respawn itself and go back to being a standby.

MANUALLY PINNING DIRECTORY TREES TO A PARTICULAR RANK

In multiple active metadata server configurations, a balancer runs which works to spread metadata load evenly across the cluster. This usually works well enough for most users but sometimes it is desirable to override the dynamic balancer with explicit mappings of metadata to particular ranks. This can allow the administrator or users to evenly spread application load or limit impact of users’ metadata requests on the entire cluster.

The mechanism provided for this purpose is called an `export pin`, an extended attribute of directories. The name of this extended attribute is `ceph.dir.pin`. Users can set this attribute using standard commands:

```
setfattr -n ceph.dir.pin -v 2 path/to/dir
```

The value of the extended attribute is the rank to assign the directory subtree to. A default value of `-1` indicates the directory is not pinned.

A directory’s export pin is inherited from its closest parent with a set export pin. In this way, setting the export pin on a directory affects all of its children. However, the parents pin can be overriden by setting the child directory’s export pin. For example:

```
mkdir -p a/b
# "a" and "a/b" both start without an export pin set
setfattr -n ceph.dir.pin -v 1 a/
# a and b are now pinned to rank 1
setfattr -n ceph.dir.pin -v 0 a/b
# a/b is now pinned to rank 0 and a/ and the rest of its children are still pinned to rank 1
```