TROUBLESHOOTING MONITORS

When a cluster encounters monitor-related troubles there's a tendency to panic, and some times with good reason. You should keep in mind that losing a monitor, or a bunch of them, don't necessarily mean that your cluster is down, as long as a majority is up, running and with a formed quorum. Regardless of how bad the situation is, the first thing you should do is to calm down, take a breath and try answering our initial troubleshooting script.

INITIAL TROUBLESHOOTING

Are the monitors running?

First of all, we need to make sure the monitors are running. You would be amazed by how often people forget to run the monitors, or restart them after an upgrade. There's no shame in that, but let's try not losing a couple of hours chasing an issue that is not there.

Are you able to connect to the monitor's servers?

Doesn't happen often, but sometimes people do have `iptables` rules that block accesses to monitor servers or monitor ports. Usually leftovers from monitor stress-testing that were forgotten at some point. Try ssh'ing into the server and, if that succeeds, try connecting to the monitor's port using you tool of choice (telnet, nc,...).

Does ceph -s run and obtain a reply from the cluster?

If the answer is yes then your cluster is up and running. One thing you can take for granted is that the monitors will only answer to a `status` request if there is a formed quorum.

If ceph `-s` blocked however, without obtaining a reply from the cluster or showing a lot of `fault` messages, then it is likely that your monitors are either down completely or just a portion is up – a portion that is not enough to form a quorum (keep in mind that a quorum if formed by a majority of monitors).

What if ceph -s doesn't finish?

If you haven't gone through all the steps so far, please go back and do.

For those running on Emperor 0.72-rc1 and forward, you will be able to contact each monitor individually asking them for their status, regardless of a quorum being formed. This can be achieved using `ceph ping mon.ID`, ID being the monitor's identifier. You should perform this for each monitor in the cluster. In section Understanding mon_status we will explain how to interpret the output of this command.

For the rest of you who don't tread on the bleeding edge, you will need to ssh into the server and use the monitor's admin socket. Please jump to Using the monitor's admin socket.

For other specific issues, keep on reading.

USING THE MONITOR'S ADMIN SOCKET

The admin socket allows you to interact with a given daemon directly using a Unix socket file. This file can be found in your monitor's `run` directory. By default, the admin socket will be kept in `/var/run/ceph/ceph-mon.ID.asok` but this can vary if you defined it otherwise. If you don't find it there, please check your `ceph.conf` for an alternative path or run:

```
ceph-conf --name mon.ID --show-config-value admin_socket
```

Please bear in mind that the admin socket will only be available while the monitor is running. When the monitor is properly shutdown, the admin socket will be removed. If however the monitor is not running and the admin socket still persists, it is likely that the monitor was improperly shutdown. Regardless, if the monitor is not running, you will not be able to use the admin socket, with ceph likely returning `Error 111: Connection Refused`.

Accessing the admin socket is as simple as telling the ceph tool to use the `asok` file. In pre-Dumpling Ceph, this can be achieved by:

```
ceph --admin-daemon /var/run/ceph/ceph-mon.<id>.asok <command>
```

while in Dumpling and beyond you can use the alternate (and recommended) format:

```
ceph daemon mon.<id> <command>
```

Using `help` as the command to the ceph tool will show you the supported commands available through the admin socket. Please take a look at `config get`, `config show`, `mon_status` and `quorum_status`, as those can be enlightening when troubleshooting a monitor.

UNDERSTANDING MON_STATUS

`mon_status` can be obtained through the ceph tool when you have a formed quorum, or via the admin socket if you don't. This command will output a multitude of information about the monitor, including the same output you would get with `quorum_status`.

Take the following example of `mon_status`:

```
{ "name": "c",
  "rank": 2,
  "state": "peon",
  "election_epoch": 38,
  "quorum": [
        1,
        2],
  "outside_quorum": [],
  "extra_probe_peers": [],
  "sync_provider": [],
  "monmap": { "epoch": 3,
      "fsid": "5c4e9d53-e2e1-478a-8061-f543f8be4cf8",
      "modified": "2013-10-30 04:12:01.945629",
      "created": "2013-10-29 14:14:41.914786",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6795\/0"}]}}
```

A couple of things are obvious: we have three monitors in the monmap (*a*, *b* and *c*), the quorum is formed by only two monitors, and *c* is in the quorum as a *peon*.

Which monitor is out of the quorum?

> The answer would be **a**.

Why?

> Take a look at the `quorum` set. We have two monitors in this set: *1* and *2*. These are not monitor names. These are monitor ranks, as established in the current monmap. We are missing the monitor with rank 0, and according to the monmap that would be `mon.a`.

By the way, how are ranks established?

> Ranks are (re)calculated whenever you add or remove monitors and follow a simple rule: the **greater** the `IP:PORT` combination, the **lower** the rank is. In this case, considering that `127.0.0.1:6789` is lower than all the remaining `IP:PORT` combinations, `mon.a` has rank 0.

MOST COMMON MONITOR ISSUES

HAVE QUORUM BUT AT LEAST ONE MONITOR IS DOWN

When this happens, depending on the version of Ceph you are running, you should be seeing something similar to:

```
$ ceph health detail
```

```
[snip]
mon.a (rank 0) addr 127.0.0.1:6789/0 is down (out of quorum)
```

How to troubleshoot this?

First, make sure `mon.a` is running.

Second, make sure you are able to connect to `mon.a`'s server from the other monitors' servers. Check the ports as well. Check `iptables` on all your monitor nodes and make sure you are not dropping/rejecting connections.

If this initial troubleshooting doesn't solve your problems, then it's time to go deeper.

First, check the problematic monitor's `mon_status` via the admin socket as explained in Using the monitor's admin socket and Understanding mon_status.

Considering the monitor is out of the quorum, its state should be one of `probing`, `electing` or `synchronizing`. If it happens to be either `leader` or peon, then the monitor believes to be in quorum, while the remaining cluster is sure it is not; or maybe it got into the quorum while we were troubleshooting the monitor, so check you `ceph -s` again just to make sure. Proceed if the monitor is not yet in the quorum.

What if the state is `probing`?

This means the monitor is still looking for the other monitors. Every time you start a monitor, the monitor will stay in this state for some time while trying to find the rest of the monitors specified in the `monmap`. The time a monitor will spend in this state can vary. For instance, when on a single-monitor cluster, the monitor will pass through the probing state almost instantaneously, since there are no other monitors around. On a multi-monitor cluster, the monitors will stay in this state until they find enough monitors to form a quorum – this means that if you have 2 out of 3 monitors down, the one remaining monitor will stay in this state indefinitively until you bring one of the other monitors up.

If you have a quorum, however, the monitor should be able to find the remaining monitors pretty fast, as long as they can be reached. If your monitor is stuck probing and you have gone through with all the communication troubleshooting, then there is a fair chance that the monitor is trying to reach the other monitors on a wrong address. `mon_status` outputs the `monmap` known to the monitor: check if the other monitor's locations match reality. If they don't, jump to Recovering a Monitor's Broken monmap; if they do, then it may be related to severe clock skews amongst the monitor nodes and you should refer to Clock Skews first, but if that doesn't solve your problem then it is the time to prepare some logs and reach out to the community (please refer to Preparing your logs on how to best prepare your logs).

What if state is `electing`?

This means the monitor is in the middle of an election. These should be fast to complete, but at times the monitors can get stuck electing. This is usually a sign of a clock skew among the monitor nodes; jump to Clock Skews for more infos on that. If all your clocks are properly synchronized, it is best if you prepare some logs and reach out to the community. This is not a state that is likely to persist and aside from (*really*) old bugs there is not an obvious reason besides clock skews on why this would happen.

What if state is `synchronizing`?

This means the monitor is synchronizing with the rest of the cluster in order to join the quorum. The synchronization process is as faster as smaller your monitor store is, so if you have a big store it may take a while. Don't worry, it should be finished soon enough.

However, if you notice that the monitor jumps from `synchronizing` to `electing` and then back to `synchronizing`, then you do have a problem: the cluster state is advancing (i.e., generating new maps) way too fast for the synchronization process to keep up. This used to be a thing in early Cuttlefish, but since then the synchronization process was quite refactored and enhanced to avoid just this sort of behavior. If this happens in later versions let us know. And bring some logs (see Preparing your logs).

What if state is `leader` or peon?

This should not happen. There is a chance this might happen however, and it has a lot to do with clock skews – see Clock Skews. If you are not suffering from clock skews, then please prepare your logs (see Preparing your logs) and reach out to us.

RECOVERING A MONITOR'S BROKEN MONMAP

This is how a `monmap` usually looks like, depending on the number of monitors:

```
epoch 3
fsid 5c4e9d53-e2e1-478a-8061-f543f8be4cf8
last_changed 2013-10-30 04:12:01.945629
created 2013-10-29 14:14:41.914786
0: 127.0.0.1:6789/0 mon.a
1: 127.0.0.1:6790/0 mon.b
2: 127.0.0.1:6795/0 mon.c
```

This may not be what you have however. For instance, in some versions of early Cuttlefish there was this one bug that could cause your `monmap` to be nullified. Completely filled with zeros. This means that not even `monmaptool` would be able to read it because it would find it hard to make sense of only-zeros. Some other times, you may end up with a monitor with a severely outdated monmap, thus being unable to find the remaining monitors (e.g., say `mon.c` is down; you add a new monitor `mon.d`, then remove `mon.a`, then add a new monitor `mon.e` and remove `mon.b`; you will end up with a totally different monmap from the one `mon.c` knows).

In this sort of situations, you have two possible solutions:

Scrap the monitor and create a new one

> You should only take this route if you are positive that you won't lose the information kept by that monitor; that you have other monitors and that they are running just fine so that your new monitor is able to synchronize from the remaining monitors. Keep in mind that destroying a monitor, if there are no other copies of its contents, may lead to loss of data.

Inject a monmap into the monitor

> Usually the safest path. You should grab the monmap from the remaining monitors and inject it into the monitor with the corrupted/lost monmap.
>
> These are the basic steps:
>
> 1. Is there a formed quorum? If so, grab the monmap from the quorum:
>
> ```
> $ ceph mon getmap -o /tmp/monmap
> ```
>
> 2. No quorum? Grab the monmap directly from another monitor (this assumes the monitor you are grabbing the monmap from has id ID-FOO and has been stopped):
>
> ```
> $ ceph-mon -i ID-FOO --extract-monmap /tmp/monmap
> ```
>
> 3. Stop the monitor you are going to inject the monmap into.
> 4. Inject the monmap:
>
> ```
> $ ceph-mon -i ID --inject-monmap /tmp/monmap
> ```
>
> 5. Start the monitor
>
> Please keep in mind that the ability to inject monmaps is a powerful feature that can cause havoc with your monitors if misused as it will overwrite the latest, existing monmap kept by the monitor.

CLOCK SKEWS

Monitors can be severely affected by significant clock skews across the monitor nodes. This usually translates into weird behavior with no obvious cause. To avoid such issues, you should run a clock synchronization tool on your monitor nodes.

What's the maximum tolerated clock skew?

> By default the monitors will allow clocks to drift up to `0.05 seconds`.

Can I increase the maximum tolerated clock skew?

> This value is configurable via the `mon-clock-drift-allowed` option, and although you *CAN* it doesn't mean you *SHOULD*. The clock skew mechanism is in place because clock skewed monitor may not properly behave. We, as developers and QA afficcionados, are comfortable with the current default value, as it will alert the user before the monitors get out hand. Changing this value without testing it first may cause unforeseen effects on the stability of

the monitors and overall cluster healthiness, although there is no risk of dataloss.

How do I know there's a clock skew?

The monitors will warn you in the form of a `HEALTH_WARN`. `ceph health detail` should show something in the form of:

```
mon.c addr 10.10.0.1:6789/0 clock skew 0.08235s > max 0.05s (latency 0.0045s)
```

That means that `mon.c` has been flagged as suffering from a clock skew.

What should I do if there's a clock skew?

Synchronize your clocks. Running an NTP client may help. If you are already using one and you hit this sort of issues, check if you are using some NTP server remote to your network and consider hosting your own NTP server on your network. This last option tends to reduce the amount of issues with monitor clock skews.

CLIENT CAN'T CONNECT OR MOUNT

Check your IP tables. Some OS install utilities add a REJECT rule to `iptables`. The rule rejects all clients trying to connect to the host except for `ssh`. If your monitor host's IP tables have such a REJECT rule in place, clients connecting from a separate node will fail to mount with a timeout error. You need to address `iptables` rules that reject clients trying to connect to Ceph daemons. For example, you would need to address rules that look like this appropriately:

```
REJECT all -- anywhere anywhere reject-with icmp-host-prohibited
```

You may also need to add rules to IP tables on your Ceph hosts to ensure that clients can access the ports associated with your Ceph monitors (i.e., port 6789 by default) and Ceph OSDs (i.e., 6800 through 7300 by default). For example:

```
iptables -A INPUT -m multiport -p tcp -s {ip-address}/{netmask} --dports 6789,6800:7300 -j AC
```

MONITOR STORE FAILURES

SYMPTOMS OF STORE CORRUPTION

Ceph monitor stores the cluster map in a key/value store such as LevelDB. If a monitor fails due to the key/value store corruption, following error messages might be found in the monitor log:

```
Corruption: error in middle of record
```

or:

```
Corruption: 1 missing files; e.g.: /var/lib/ceph/mon/mon.0/store.db/1234567.ldb
```

RECOVERY USING HEALTHY MONITOR(S)

If there is any survivers, we can always replace the corrupted one with a new one. And after booting up, the new joiner will sync up with a healthy peer, and once it is fully sync'ed, it will be able to serve the clients.

RECOVERY USING OSDS

But what if all monitors fail at the same time? Since users are encouraged to deploy at least three monitors in a Ceph cluster, the chance of simultaneous failure is rare. But unplanned power-downs in a data center with improperly configured disk/fs settings could fail the underlying filesystem, and hence kill all the monitors. In this case, we can recover the monitor store with the information stored in OSDs.:

```
ms=/tmp/mon-store
mkdir $ms
# collect the cluster map from OSDs
for host in $hosts; do
  rsync -avz $ms user@host:$ms
  rm -rf $ms
  ssh user@host <<EOF
    for osd in /var/lib/osd/osd-*; do
      ceph-objectstore-tool --data-path \$osd --op update-mon-db --mon-store-path $ms
    done
  EOF
  rsync -avz user@host:$ms $ms
done
# rebuild the monitor store from the collected map, if the cluster does not
# use cephx authentication, we can skip the following steps to update the
# keyring with the caps, and there is no need to pass the "--keyring" option.
# i.e. just use "ceph-monstore-tool /tmp/mon-store rebuild" instead
ceph-authtool /path/to/admin.keyring -n mon. \
  --cap mon 'allow *'
ceph-authtool /path/to/admin.keyring -n client.admin \
  --cap mon 'allow *' --cap osd 'allow *' --cap mds 'allow *'
ceph-monstore-tool /tmp/mon-store rebuild -- --keyring /path/to/admin.keyring
# backup corrupted store.db just in case
mv /var/lib/ceph/mon/mon.0/store.db /var/lib/ceph/mon/mon.0/store.db.corrupted
mv /tmp/mon-store/store.db /var/lib/ceph/mon/mon.0/store.db
chown -R ceph:ceph /var/lib/ceph/mon/mon.0/store.db
```

The steps above

1. collect the map from all OSD hosts,
2. then rebuild the store,
3. fill the entities in keyring file with appropriate caps
4. replace the corrupted store on `mon.0` with the recovered copy.

KNOWN LIMITATIONS

Following information are not recoverable using the steps above:

- **some added keyrings**: all the OSD keyrings added using `ceph auth add` command are recovered from the OSD's copy. And the `client.admin` keyring is imported using `ceph-monstore-tool`. But the MDS keyrings and other keyrings are missing in the recovered monitor store. You might need to re-add them manually.
- **pg settings**: the `full ratio` and `nearfull ratio` settings configured using `ceph pg set_full_ratio` and `ceph pg set_nearfull_ratio` will be lost.
- **MDS Maps**: the MDS maps are lost.

EVERYTHING FAILED! NOW WHAT?

REACHING OUT FOR HELP

You can find us on IRC at #ceph and #ceph-devel at OFTC (server irc.oftc.net) and on `ceph-devel@vger.kernel.org` and `ceph-users@lists.ceph.com`. Make sure you have grabbed your logs and have them ready if someone asks: the faster the interaction and lower the latency in response, the better chances everyone's time is optimized.

PREPARING YOUR LOGS

Monitor logs are, by default, kept in `/var/log/ceph/ceph-mon.FOO.log*`. We may want them. However, your logs may not have the necessary information. If you don't find your monitor logs at their default location, you can check where they should be by running:

```
ceph-conf --name mon.FOO --show-config-value log_file
```

The amount of information in the logs are subject to the debug levels being enforced by your configuration files. If you have not enforced a specific debug level then Ceph is using the default levels and your logs may not contain important information to track down you issue. A first step in getting relevant information into your logs will be to raise debug levels. In this case we

will be interested in the information from the monitor. Similarly to what happens on other components, different parts of the monitor will output their debug information on different subsystems.

You will have to raise the debug levels of those subsystems more closely related to your issue. This may not be an easy task for someone unfamiliar with troubleshooting Ceph. For most situations, setting the following options on your monitors will be enough to pinpoint a potential source of the issue:

```
debug mon = 10
debug ms = 1
```

If we find that these debug levels are not enough, there's a chance we may ask you to raise them or even define other debug subsystems to obtain infos from – but at least we started off with some useful information, instead of a massively empty log without much to go on with.

DO I NEED TO RESTART A MONITOR TO ADJUST DEBUG LEVELS?

No. You may do it in one of two ways:

You have quorum

Either inject the debug option into the monitor you want to debug:

```
ceph tell mon.FOO config set debug_mon 10/10
```

or into all monitors at once:

```
ceph tell mon.* config set debug_mon 10/10
```

No quourm

Use the monitor's admin socket and directly adjust the configuration options:

```
ceph daemon mon.FOO config set debug_mon 10/10
```

Going back to default values is as easy as rerunning the above commands using the debug level `1/10` instead. You can check your current values using the admin socket and the following commands:

```
ceph daemon mon.FOO config show
```

or:

```
ceph daemon mon.FOO config get 'OPTION_NAME'
```

REPRODUCED THE PROBLEM WITH APPROPRIATE DEBUG LEVELS. NOW WHAT?

Ideally you would send us only the relevant portions of your logs. We realise that figuring out the corresponding portion may not be the easiest of tasks. Therefore, we won't hold it to you if you provide the full log, but common sense should be employed. If your log has hundreds of thousands of lines, it may get tricky to go through the whole thing, specially if we are not aware at which point, whatever your issue is, happened. For instance, when reproducing, keep in mind to write down current time and date and to extract the relevant portions of your logs based on that.

Finally, you should reach out to us on the mailing lists, on IRC or file a new issue on the tracker.