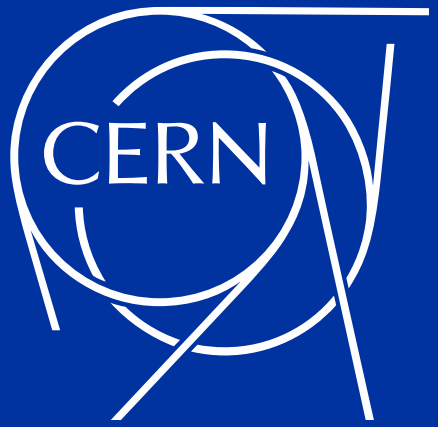

EOSHOME Operations Bits and Pieces

Roberto VALVERDE (IT-ST, CERN)

02/03/2021

EOSHOME Operations - Bits and Pieces

- Reduce EOS FST system disk load
  - EOS logs
  - LevelDB
- EOS geo-scheduler use cases
  - Adding new capacity
  - Targeted draining
- Moving EOS config to QuarkDB

Reducing EOS FST system disk load

- Issue in 2020
- Symptoms:
  - EOS slow down
  - Node responsiveness problem
    - Slow system performance
    - Puppet time outs
  - The affected nodes had system disk usage always at 100%
  - Only disk servers with system disk on spinners were affected.

Reducing EOS FST system disk load (II)

- **Go SSD**
- **Reduce log verbosity for EOS (/var/log/eos)**

```
eos debug info *                set MGM & all FSTs into debug mode 'info'
eos debug err /eos/*/fst        set all FSTs into debug mode 'info'
eos debug crit /eos/*/fst       set all FSTs into debug mode 'crit'
```

(*) Configuration not persisted

- **Review** logrotate **configuration**
  - **Archive compressed logs to another location**
  - **Or move log storage to another disk**
  - **In older EOS versions (<4.8.0) , the FST daemons could take some time to start (or even fail) if this folder is very populated.**

Reducing EOS FST system disk load (III)

- **LevelDB store FST file metadata and generates high number disk accesses.**
  - **Specially noticeable on multi-fst setups**
  - **Move** /var/eos/md **to a different disk**
  - **Since EOS** v4.8.4**, LevelDB can be place on the actual data disk**

```
EOS_FST_FMD_ON_DATA_DISK=1
```

EOS geo-scheduler use cases

- When EOS instance is quite full, adding a new disk server can be problematic:
  - Most free space will be on a single node, and it will be the preferred target for the placement.
  - If the workflow is intense, node can get overloaded.
  - Specially problematic on workflows where a lot of data is written and read right after.

EOS geo-scheduler use cases (II)

- **Do not wait the instance to be full in order to add new capacity.**
- **Add several new disk servers at the same time**
- **Use EOS** geoscheduler**:**
  - **The EOS scheduler is a core component which decides on which filesystems to place or access files based on geotags.**

    ```
    EOS_GEOTAG="0513::R::0050::CQ18"
    ```

  - **You can create policies to allow / disable different operations based on node geotag.**
  - **While adding a new node, we can disable the file placement before adding it to the main data pool.**

    ```
    eos geosched disabled add 0513::R::0050::CQ18 plct \*
    ```

EOS geo-scheduler use cases (III)

```
eos geosched disabled add 0513::R::0050::CQ18 plct \*
```

- **Use this in combination with the EOS** balancer
  - **Disabling placement won't affect other internal EOS systems, like draining, balancing, converter…**
  - **Balancing will distribute free space across different disk servers.**
  - **As balancing selects replicas randomly, no new files -> new server correlation.**
- **Maybe a good moment to review balancing configuration:**

```
eos space config <space-name> space.balancer.node.rate=<MB/s>                : configure the nominal
transfer bandwidth per running transfer on a node [ default=25 (MB/s)   ]
eos space config <space-name> space.balancer.node.ntx=<#>                    : configure the number of
parallel balancing transfers per node           [ default=2 (streams) ]
```

EOS geo-scheduler use cases (IV)

- **Review balancing traffic:**

```
[root@eoshome-ns-i02-02 (mgm:master mq:master) ~]$ eos io stat -x
┌──┬──────────────┬──────┬──────┬──────┬──────┐
│io│   application│  1min│  5min│    1h│   24h│
└──┴──────────────┴──────┴──────┴──────┴──────┘
out          eos/balancing   1.69 G  13.43 G 281.28 G   6.71 T
```

```
[root@eoshome-ns-i02-02 (mgm:master mq:master) ~]$ eos group ls balancing
┌────────┬──────────┬──────┬──────┬──────────┬──────────┬──────────┬─────────┬──────┐
│type    │      name│status│N(fs) │dev(filled)│avg(filled)│sig(filled)│balancing│bal-shd│
└────────┴──────────┴──────┴──────┴──────────┴──────────┴──────────┴─────────┴──────┘
 groupview default.0    on    11       38.05      86.32      13.73 balancing      9
 groupview default.1    on    11       39.02      85.83      14.11 balancing      3
```

- **Once the free space is better distributed on different nodes, we can revert the rule:**

```
eos geosched disabled rm 0513::R::0050::CQ18 plct \*
```

EOS geo-scheduler use cases: Draining

- In 2019 we perform a draining campaign of the storage in Wigner CERN datacenter
  - When we started the draining we realised that we had traffic going from Meyrin to Wigner nodes (?)
  - EOS uses secondary replica for replication:
    - Rep 00: Wigner (draining)
    - Rep 01: Meyrin (used to do the replication) -> The new replica could end in Wigner Datacenter again.

EOS geo-scheduler use cases (III)

- **This will "ban" the nodes under that geotag and they won't be valid for receive replicas generated via the draining.**

```
eos geosched disabled add WIG::R::0050:: plctdrain \*
```

- **This should not happen if you drain the full set of nodes at the time, as nodes in draining status are not valid targets for receiving replicas.**

```
space config <space-name> space.drainer.node.rate=<MB/s >              : configure the nominal transfer bandwidth per
running transfer on a node [ default=25 (MB/s)   ]
space config <space-name> space.drainer.node.ntx=<#>                   : configure the number of parallel draining
transfers per node            [ default=2 (streams) ]
space config <space-name> space.drainer.node.nfs=<#>                   : configure the number of max draining
filesystems per node (Valid only for central drain)  [ default=5 ]
space config <space-name> space.drainer.retries=<#>                    : configure the number of retry for the
draining process (Valid only for central drain)     [ default=1 ]
space config <space-name> space.drainer.fs.ntx=<#>                     : configure the number of parallel draining
transfers per fs (Valid only for central drain) [ default=5 ]
```

```
eos ns max_drain_threads <num>
    set the max number of threads in the drain pool, default 400, minimum 4
```

EOS configuration in QuarkDB

- File-based configuration will be dropped in EOS v5
- PROCEDURE
  - This need be run from the current master mgm of the instance:
  - Make sure that `eos-ns-inspect` is installed (version 4.7.16 or above is recommended)
  - Run the following command to export local config to QuarkDB (safe!)

```
eos-config-inspect export --source /var/eos/config/<hostname>.cern.ch/default.eoscf --members
eos<instance>-qdb:7777 --password-file <pass_file>
```

  - Now dump the exported data to cross-check that was correctly imported

```
eos-config-inspect dump --password-file <pass_file> --members eos<instance>-qdb.cern.ch:7777 | tee
default.eoscf.qdb
```

  - Diff local config with the one generated by the previous command

EOS configuration in QuarkDB (II)

- If everything seems fine, let's change the mgm configuration (/etc/xrd.cf.mgm) to use QuarkDB configuration:

```
mgmofs.cfgtype  quarkdb
```

- Stop the mgm:

```
systemctl stop eos@mgm
```

- Compare again the local config file with a new fresh dump of the config in QuarkDB.

```
eos-config-inspect dump --password-file <pass_file> --members eos<instance>-qdb.cern.ch:7777 | tee
default.eoscf.qdb.2
```

- If there are changes, run again the export command but adding the `--overwrite` flag. If not, skip the following step.

```
eos-config-inspect export --source /var/eos/config/<hostname>.cern.ch/default.eoscf --members
eos<instance>-qdb:7777 --password-file <pass_file> --overwrite
```

- If there are no changes, means that we have already the latest configuration on quarkdb. We need just to start the mgm.

```
systemctl start eos@mgm
```

Demo

EOSHOME Operations: Bits and Pieces

- More info about EOS Scheduler in EOS Documentation
- More info about EOS Draining system in EOS Documentation
- More info about QuarkDB Documentation

Thank you!

home.cern