CERN

Improving EOS availability through early error detection and recovery - FSCK

Elvin Sindrilaru
on behalf of the **EOS team**

Outline

- Why we need fsck?
- Design of the new fsck subsystem
  - MGM components
  - FST components
- Types of errors detected
- Future developments

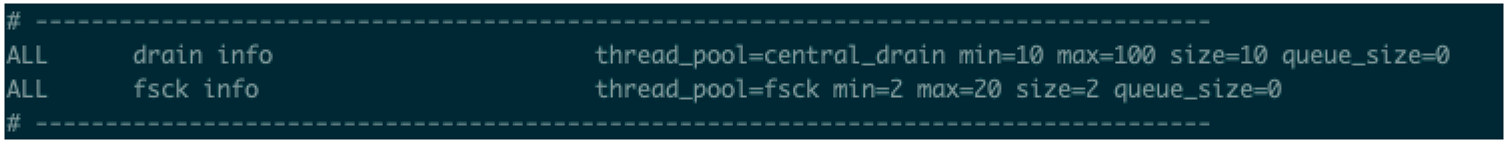
Why we need fsck?

- **Detect corruptions** on the disk server
- **Protect** against "silent" corruptions or hardware issues
- Maintain consistency between the **namespace view** and the **file system view**
- Satisfy the "**service level agreement**"
    - Maintain required data redundancy
    - Avoid data loss due to preventable error scenarios

FSCK MGM components

- Separated into two stages (different threads)
  - **Collection** of errors
  - **Repairing** errors

- Thread pool dedicated to FSCK repair jobs



```
# ------------------------------------------------------------------------------------
ALL      drain info                        thread_pool=central_drain min=10 max=100 size=10 queue_size=0
ALL      fsck info                         thread_pool=fsck min=2 max=20 size=2 queue_size=0
# ------------------------------------------------------------------------------------
```

- Simplified client console command
  - Enable/disable error collection
  - Enable/disable repair thread
  - Trigger repair for individual files/ids
  - Generate text/json report for errors

Monitoring FSCK at the MGM

- Activity in the dedicated thread pool - “***eos ns***” command

- “***eos ns stat***” counters for Fsck jobs
    - # Fsck repair jobs started
    - # Fsck repair jobs successful
    - # Fsck repair jobs failed

- Logs for the individual jobs are in the main log file and also in “**DrainJob.log**”

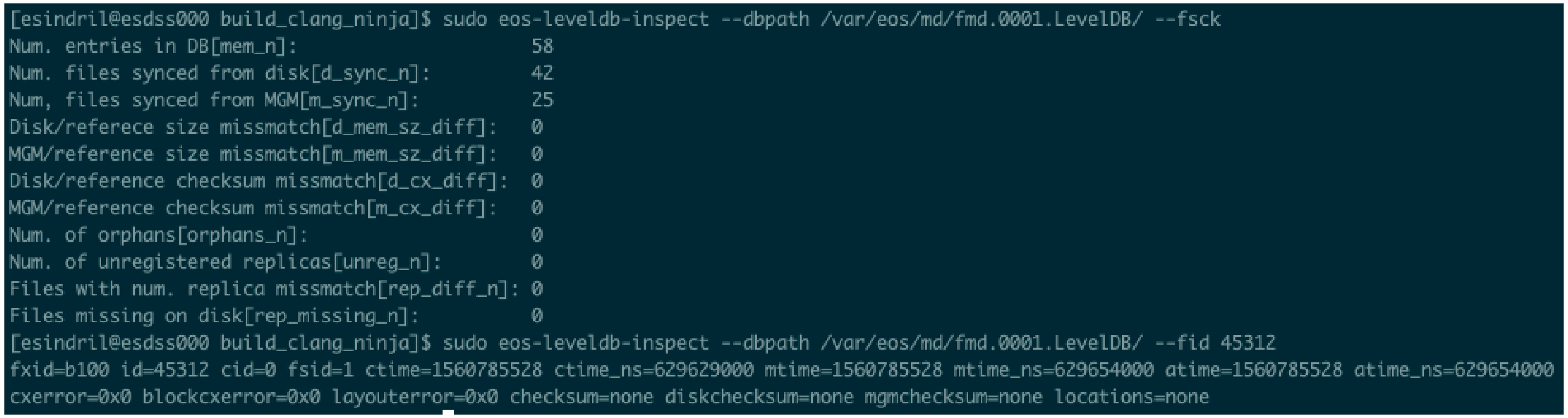
- Most Fsck repair jobs -> Drain Jobs

What info is saved where and when?

- After a **write or update** operation on the FST the following happens:
  - Update **the local DB** with all the available info (partial – no MGM input)
  - Set (some) **extended attributes** with same info as in the local DB
  - After at least 60 seconds **sync local DB with MGM info** (all the mgm… fields)

- Some info is *redundant* – we rely only on the one stored in the local DB

- New tool to easily dump info from the local DB:
  - **eos-leveldb-inspect**



```
[esindril@esdss000 build_clang_ninja]$ sudo eos-leveldb-inspect --dbpath /var/eos/md/fmd.0001.LevelDB/ --fsck
Num. entries in DB[mem_n]:                  58
Num. files synced from disk[d_sync_n]:      42
Num, files synced from MGM[m_sync_n]:       25
Disk/referece size missmatch[d_mem_sz_diff]:  0
MGM/reference size missmatch[m_mem_sz_diff]:  0
Disk/reference checksum missmatch[d_cx_diff]: 0
MGM/reference checksum missmatch[m_cx_diff]:  0
Num. of orphans[orphans_n]:                 0
Num. of unregistered replicas[unreg_n]:     0
Files with num. replica missmatch[rep_diff_n]: 0
Files missing on disk[rep_missing_n]:       0
[esindril@esdss000 build_clang_ninja]$ sudo eos-leveldb-inspect --dbpath /var/eos/md/fmd.0001.LevelDB/ --fid 45312
fxid=b100 id=45312 cid=0 fsid=1 ctime=1560785528 ctime_ns=629629000 mtime=1560785528 mtime_ns=629654000 atime=1560785528 atime_ns=629654000
cxerror=0x0 blockcxerror=0x0 layouterror=0x0 checksum=none diskchecksum=none mgmchecksum=none locations=none
```

File info stored in local DB (FmdHelper)

- fid – file id
- cid – container id
- uid – user id
- gid – group id
- fsid – file system id
- ctime – change time
- mtime – modification time
- atime – access time
- size – reference size
- mgmsize – size on MGM
- checksum – reference checksum
- mgmchecksum – checksum on MGM
- lid – layout id
- locations – set of fsids for replicas
- **checktime – timestamp of last scan (updated by scan)**
- **disksize – size on disk (updated by scan)**
- **diskchecksum – checksum of file on disk (updated by scan)**
- **filecxerror – flag for file checksum errors**
- **blockcxerror – flag for block checksum errors**
- **layouterror – flag for various other inconsistencies**

FSCK FST components

- Scanning done from different perspectives
  - **Disk scanning** – ensure file system -> namespace consistency
    - Update info about file in the local LevelDB
    - One thread per file system
      - Runs every 4 hours (**scan_disk_interval**)
      - Checks file not verified in the last 7 days (**scaninterval**)
  - **Namespace scanning** – ensure namespace -> file system consistency
    - Done to detect **missing** files
    - Connect directly to QuarkDB and minimize data requests
    - Runs every 3 days (**scan_ns_interval**)
- All parameters can be configured per file-system using "***eos fs config***"

Layouterror

- Can be one of the following:
  - **kOrphan** – there is no entry at the MGM concerning the file. These get moved to **.eosorphans** on the current mount point.
  - **kUnregistered** – file exists at the MGM but this replica is not in the list of locations.
  - **kReplicaWrong** – the nominal number of replicas given by the layout is different from the number of valid replicas in the *locations* vector. Can be more or less …
  - **kMissing** – the MGM thinks there is replica but actually there is no file on disk.

Errors on FST and their resolution

Decreasing priority

- **d_mem_sz_diff** – disk and reference size mismatch – fixed by FsckRepairJob
- **m_mem_sz_diff** – MGM and reference size mismatch – fixed by inspecting all the replicas or saved for manual inspection
- **d_cx_diff** – disk and reference checksum mismatch – fixed by FsckRepairJob
- **m_cx_diff** – MGM and reference checksum mismatch – fixed by inspecting all the replicas or saved for manual inspection
- **unreg_n** – register replica if metadata match or drop if not needed
- **rep_missing_n** – fixed by FsckRepairJob
- **rep_diff_n** – fixed by dropping replicas or creating new ones through FsckRepairJob
- **orphans_n** – no action at the MGM

Transient errors detected at the MGM

- Errors due to file systems not being online or in a bad state
- Detecting these puts a lot of **pressure on the QDB namespace** – requires full scan of files on concerned FSTs
- Examples:
  - **rep_offline** – files with replicas offline
  - **rep_diff_n** – represents a superset of the ones reported by FSTs
  - **file_offline** – files for which all replicas are offline
- These were fixed by doing an adjust replica operation
- This is **equivalent to automatically triggering the drain of a file system** which has been offline for a certain period of time.

Plans for the future ...

- The **FmdBase protobuf object** will no longer be stored in the local DB but as an **extended attribute on the file**
  - Reduce info duplication
  - Eliminate a possible inconsistency between disk and local DB
  - Possibility to use compression (lzstd)
- New fsck available since **eos-4.6.0**