FSCK to the rescue

Elvin Sindrilaru

on behalf of the **EOS team**

25.04.2023

Outline

- Why we need fsck?
- Scanner functionality and configuration
- Types of errors detected
- Fsck components
  - New architecture using QuarkDB backend
  - Collection
  - Repair

Why we need fsck?

- **Fsck** mechanism is designed to **automatically detect and repair** different types of errors
- Maintains **consistency** between the **namespace** and the **file system view**
- Fsck relies on the information collected by the **ScanDir functionality**
- Crucial mechanism for:
  - **Detection of silent corruptions**
  - **Hardware failures**
  - **Software bugs**
  - **Human errors**
- **Manual** tracking and repair **not scalable** for O(1B) files
- **Role of fsck** - make an operator's life easier by **reducing** the amount of **manual work**

ScanDir functionality and configuration

- **One thread per file system** scanning all the files on the mount-point
- Running in **lowest I/O priority (7) in the best-effort** scheduling class
- **Configurable behavior per FS** based on the following parameters:

Parameter	Description
scaninterval	File entry rescan interval (default 7 days)
scanrate	Max IO scan rate per fs (50 MB/s)
scan_disk_interval	Disk thread scan interval (4 hours)
scan_ns_interval	Namespace thread scan interval (3 days)
scan_ns_rate	Max NS scan rate (50 entries/sec)

- Can be completely disabled if `scaninterval` set to 0 – **not advisable, but useful in some cases!**
- **Collects and publishes inconsistencies** in LevelDB or directly in QuarkDB

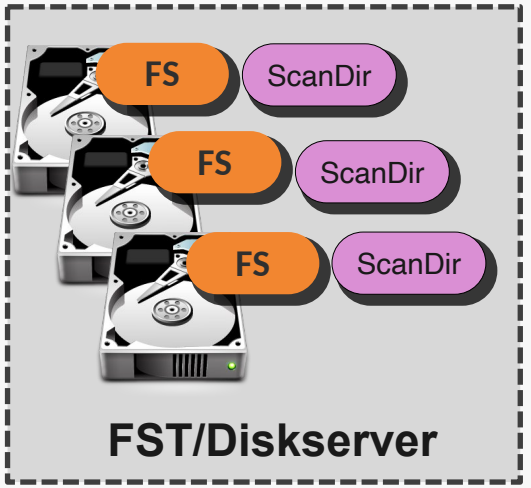
Disk and namespace consistency

- Scanning trying to reconcile the split between
  - **Metadata/namespace** – stored at the MGM (actually in QuarkDB)
  - **Physical data on disk** – stored in the FSTs

- **Disk scanning** – performed by the **ScanDir** thread

- **Namespace scanning** – separate thread
  - Check that what the namespace expects to be on **disk matches reality**
  - **Light-weight activity** just checking that the entries exist on disk
  - Not accessing the FileMD object from the namespace

- Output of all scanning activities is collected in the **FSCK status**



Collected information

Field	Description
fid	File identifier
cid	(Parent) container identifier
uid/gid	User/group ID
fsid	File system ID
ctime/mtime/atime	Change/modification/access time
size	Reference size (when written)
mgmsize	Size in the namespace (MGM)
checksum	Reference checksum (when written)
mgmchecksum	Checksum in the namespace (MGM)
lid	Layout ID
locations	Set of FS ids of other stripes/replicas
checktime	Last scan timestamp
disksize	Size on disk
diskchecksum	Checksum of file on disk
filecxerror	Flag for file checksum error
blockcxerror	Flag for block checksum error
layouterror	Flag for other types of inconsistencies

Updated by ScanDir

Fsck errors

“layouterror” category

Sub-type	Description
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	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	MGM thinks there is replica but actually there is no file on disk.

Move from LevelDB to xattrs

- **LevelDB** support will soon be deprecated (>= 5.2.0)
- FSTs becomes **stateless** – opens the possibility of using a different back-end i.e CEPH
- Open/close performance is more **stable and reliable**
  - Previous jittery performance due to serialization on the LevelDB access
- **Conversion is performed automatically** when restarting the FSTs
  - Configuration directive in `/etc/xrd.cf.fst`:
    - `fstofs.filemd_handler attr`
- Converted file systems have hidden `.eosattrconverted` on the root of the mount-point
- **Conversion speed** is around **100k entries / minute**
- **eos-fmd-tool** – used to perform targeted conversions or inspect the info about a file
  - Replacement for `eos-leveldb-inspect`

eos-fmd-tool example output

```
eos-fmd-tool inspect --path /data58/0002adf5/68ab417f

fid: 1756053887
cid: 136404885
fsid: 16459
ctime: 1680780555
ctime_ns: 191810910
mtime: 1680751427
mtime_ns: 678276538
atime: 1680780556
atime_ns: 289492000
checktime: 1680679353
size: 165332501
disksize: 165332501
mgmsize: 165332501
checksum: "2d7963ed"
diskchecksum: "2d7963ed"
mgmchecksum: "2d7963ed"
lid: 6291730
uid: 50863
gid: 1470
filecxerror: 0
blockcxerror: 0
layouterror: 0
locations: "16459,16363"
```

```
getfattr -d /data58/0002adf5/68ab417f
# file: data58/0002adf5/68ab417f
user.eos.blockcxerror="0"
user.eos.checksum=0s7WN5LQ==
user.eos.checksumtype="adler"
user.eos.filecxerror="0"
user.eos.fmd=0sCX9Bq2gAAAAAEZVfIQgAAAAAHUtAAAAlC60uZC1ezW4LNUM7LmQ9uq1tKEUMrS5kTSBMQRFVuSEtZFkVxtoJAAAAAGEVxtoJAAAA
AGkVxtoJAAAAAHIIMmQ3OTYzZWR6CDJkNzk2M2VkggEIMmQ3OTYzZWSNARIBYACVAa/GAACdAb4FAACgAQCoAQCwAQC6AQsxNjQ1OSwxNjM2Mw==
user.eos.lfn="/eos/lhcb/user/l/wwwwww/xxxxxx/yyyyy/zzzzz/Gauss.sim"
user.eos.timestamp="1680679353"
```

Fsck collection

- Display of aggregated information **collected already by the FSTs**

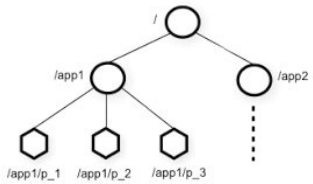
- **LevelDB implementation**
  - **MGM broadcasts request and collects responses** form FSTs using point-to-point communication
  - Requires read-lock of the LevelDB **impacting write operations**

- **Xattr implementation**
  - FSCK info published directly in QuarkDB by the FSTs
  - MGM pulls and updates this information regularly
  - No more direct contact between MGM and FSTs
  - Each fsck error category is a set containing:
    - `<file_id>:<fs_id>` entries

```
$ redis-cli -p 7777 keys "fsck*"
1) "fsck:d_cx_diff"
2) "fsck:d_mem_sz_diff"
3) "fsck:m_cx_diff"
4) "fsck:m_mem_sz_diff"
5) "fsck:orphans_n"
6) "fsck:rep_diff_n"
7) "fsck:rep_missing_n"
8) "fsck:unreg_n"
$ redis-cli -p 7777 scard "fsck:m_cx_diff"
(integer) 209
```

Example fsck collection summary

1B entries namespace




```
$ eos fsck stat
Info: collection thread status -> enabled
Info: repair thread status     -> enabled
Info: repair category          -> all
230412 12:11:31 Start error collection
230412 12:11:31 Filesystems to check: 5234
230412 12:11:31 d_cx_diff                  : 2141
230412 12:11:31 d_mem_sz_diff              : 483
230412 12:11:31 m_cx_diff                  : 209
230412 12:11:31 m_mem_sz_diff              : 1118
230412 12:11:31 orphans_n                  : 1
230412 12:11:31 rep_diff_n                 : 261
230412 12:11:31 rep_missing_n              : 39
230412 12:11:31 unreg_n                    : 93
230412 12:11:31 Finished error collection
230412 12:11:31 Next run in 10 minutes
```

Same fids can be in several categories

Fsck repair

- Repair is **driven from the MGM** and uses as input the collected FSCK inconsistencies
- Repair can **target a specific category of errors** or **all** of them
- Repair can be **triggered for individual files**
  - Requires as much information about the corruption as possible

```
$ eos fsck repair --fxid 5841a80e --fsid 14319 --error rep_missing_n
```

- All fsck repair transfers are tagged with the “eos/fsck” tag
- Internally most repair actions involve triggering a **third-party-copy operation (TPC)**
- Fsck triggered **deletions** of raw data can be **converted to move operations**
  - Use the env. variable **EOS_FST_FSCK_DELETE_BY_MOVE** on the FSTs
  - Files are stored in a hidden **.eosdeletions** to be cleaned by the operators

Fsck repair monitoring

- All fsck repair jobs run in a separated **dynamically scaling thread-pool**
- All file being handled by FSCK are **internally tracked to avoid interference** from other internal operations
- **Pool and queue parameters** modified using the `fsck config` command

```
$ eos ns stat | grep fsck
ALL      fsck info                    pool=fsck          min=2   max=20   size=20   queue_sz=0
ALL      tracker info                 tracker=fsck size=0
```

- Progress on the **overall fsck activity**:

```
$ eos ns stat | grep Fsck
all FsckRepairFailed       21.28 M     0.00     0.00     0.00     8.24
all FsckRepairStarted      21.30 M     0.00     0.00     0.00     8.25
all FsckRepairSuccessful   16.48 K     0.00     0.00     0.00     0.01
```

- Fsck command can generate **reports** for errors per file systems in **text or JSON output**

Fsck repair actions

- **FSCK** repair mechanism is as conservative as possible
- **Corruptions without a clear resolution** are left for the operator to handle

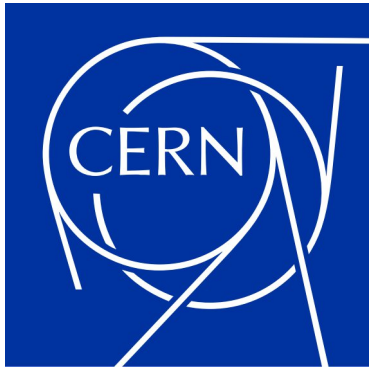
Decreasing repair priority

Error type	Resolution
blockxs_err	Only for RAIN – trigger stripe recovery
unreg_n	Register replica if metadata match or drop if not needed
rep_diff_n	Fixed by dropping excess replicas or creating new ones through FsckRepairJob
rep_missing_n	Fixe by triggering a FsckRepairJob
m_mem_sz_dif	MGM and reference size mismatch – fixed by inspecting all the replicas or saved for manual inspection
m_cx_diff	MGM and reference checksum mismatch – fixed by inspecting all the replicas or saved for manual inspection
d_mem_sz_diff	Disk and reference size mismatch – fixed by FsckRepairJob
d_cx_diff	Disk and reference checksum mismatch – fixed by FsckRepairJob

Next steps

- FSCK collection and repair now running on all EOS physics instances 🥳
- The current mechanism is **easily extensible** with new repair procedures
- Track down, isolate and fix code paths that **systematically generate inconsistencies**

Thank you! Questions? Comments?



home.cern