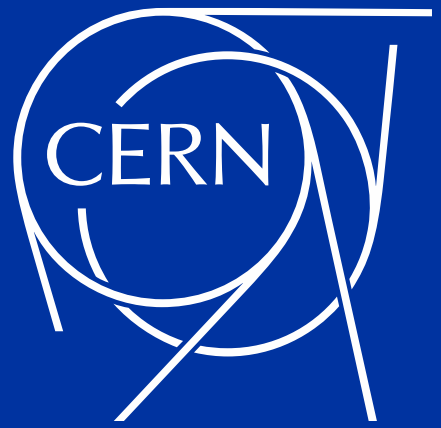


Improving FST metadata storage

Retiring LevelDBs in favor of file attribute storage

Abhishek Lekshmanan on behalf of the EOS team

Outline

- File Metadata storage
  - Motivation
  - Issues with current LevelDB
- Attribute Backend
  - Implementation
  - Deployment Configuration
  - Test Results
  - `eos-filemd-convert` tool

Introduction

File Metadata Storage

File Metadata Storage - Motivation

- Storage systems need to periodically do **File System Consistency Checks** to ensure healthy files
- EOS needs to ensure both the MGM/namespace view of a File and the replica/stripes stored on FSTs are consistent
  - FSTs do this by storing **file consistency metadata** on the FST nodes
  - Consistency metadata basically has info about the file/container and various checksums etc.
  - FSCK processes reference this metadata to check the actual on disk state of files
  - The rest of this presentation will use file metadata to mean **file consistency metadata**
- Default storage for the FSCK metadata on FSTs is LevelDBs on the metadata directory of FSTs, defaults to /var/eos/md/

```
fid: 28724711
cid: 53321
fsid: 457
ctime: 1660594128
ctime_ns: 244109250
mtime: 1660594128
mtime_ns: 608671000
atime: 1668616548
atime_ns: 286749000
checktime: 1682259962
size: 2
disksize: 2
mgmsize: 2
checksum: "00ce006c"
diskchecksum: "00ce006c"
mgmchecksum: "00ce006c"
lid: 1048850
uid: 0
gid: 0
filecxerror: 0
blockcxerror: 0
layouterror: 0
locations: "147,457"
```

Issues with LevelDB

- Used in the data path, every **open** for write call for eg. Would necessitate inserting the file consistency metadata in storage
  - Translates to levelDB transaction *for every write*
  - Logged times spent for various open subroutines, levelDB writes were quire jittery
  - **Noisy neighbors** Since LevelDBs were colocated on a single /var/ partition, we were serializing on single disk IO wrt file metadata write
- External dependency we don’t maintain
  - Optimizing LevelDB writes is a project by itself
  - Most projects moved to RocksDB and other implementations for K-V store

Attribute Backend

Motivation

- Disks no longer fully represented the state of an FST, additional state in LevelDB
- Reduce dependencies on external components
- Avoid single point of contention for filesystem IO
  - Leverage the fact that we create the file anyway, so attribute writes are cheap
  - Size of metadata quite small (<KB), easily fits in any file attribute
  - Idea was to introduce a new backend, storing FSCK metadata as attributes on the local disk itself
  - Moving to a new backend potentially opens up to other backend File Storages as well while keeping FSCK info

Implementation

- The layer committing the file metadata storage was refactored to support multiple backends
  - Attribute based backend was introduced
  - The storage layer itself was agnostic to the type of backend storage involved - made writing a converter simple
- FSTs check which metadata backend to commit to using a simple configurable (fstofs.filemd_handler) in `xrd.cf.fst`
- A converter was introduced which essentially translates the file metadata from one backend to another
- We defaulted to level DB backend, however if the attributes were configured as the backend, conversion will be triggered

Deployment Configuration

- **xrd.cf.fst** - This is the only configuration to be set

```
fst.filemd_handler: attr # or leveldb default
```

- **/etc/sysconfig/eos_env:** Additional env vars control the size of conversion queue - affects speed and memory conversions

```
EOS_FMD_GLOBAL_QUEUE_SIZE: 4,000,000
EOS_FMD_PER_FS_QUEUE_SIZE: 1,00,000
```

  - FST Launches conversions on startup, logs one line every file converted
  - Defaults are usually sane, however if you have an extremely dense node, recommended to launch one FST upgrade and try tweaking the parameters to see if the conversion speed is optimal

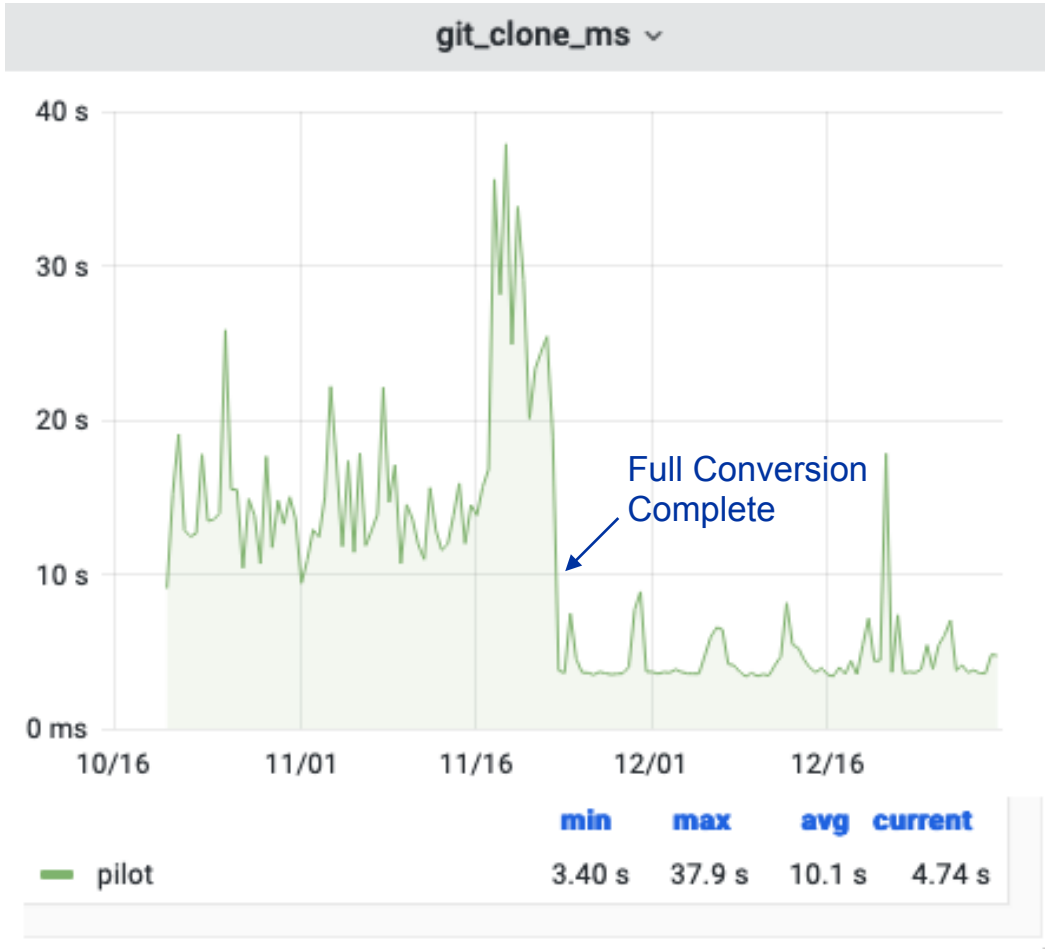
- On finishing conversions a touch file `.eosattrconverted` is placed at the root of filesystem. FST checks for this at startup to avoid converting again

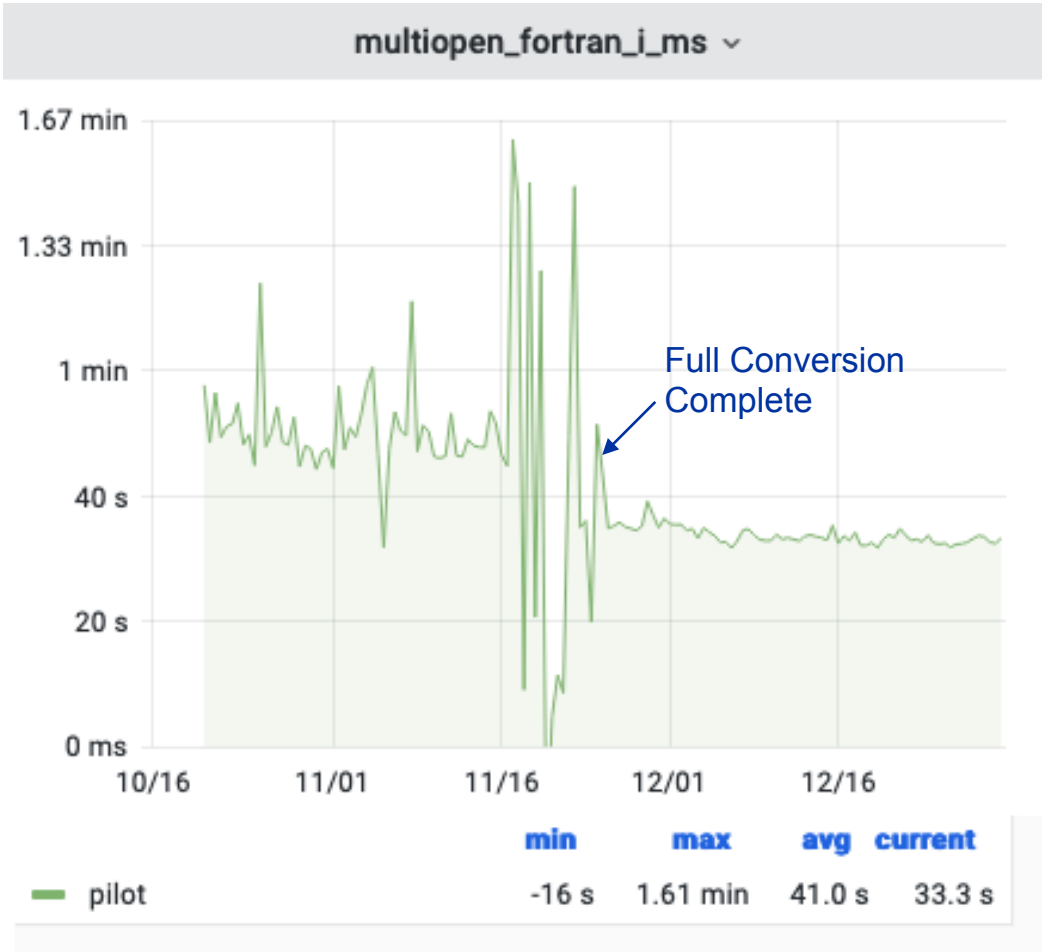
```
.. func=Convert  level=INFO source=FmdConverter:113 ... msg="conversion done" file=/data12/00005173/0c6db202, fid=0c6db202, status=1
.. func=Convert  level=INFO source=FmdConverter:113 ... msg="conversion done" file=/data31/00002f8b/07413fd9, fid=07413fd9, status=1
.. func=Convert  level=INFO source=FmdConverter:113 ... msg="conversion done" file=/data72/0000d074/1fcec798, fid=1fcec798, status=1
.. func=Convert  level=INFO source=FmdConverter:113 ... msg="conversion done" file=/data72/0000a74b/1986fcf3, fid=1986fcf3, status=1
```

Results

- Initial tests done on EOSBackup instance - nodes with 0.5 Billion files took upto 2 days
  - Conversion speed bound mainly by file count rather than disk fullness
- All **EOS for Physics** instances have moved to the new attr backend for FSCK metadata (EOSPublic ongoing)
  - Most instances with a few tens of million files took under a couple of hours for conversion per FST
- Microtests which do metadata heavy operations like git clone and filesystem walk saw less jitters after moving to the new backend

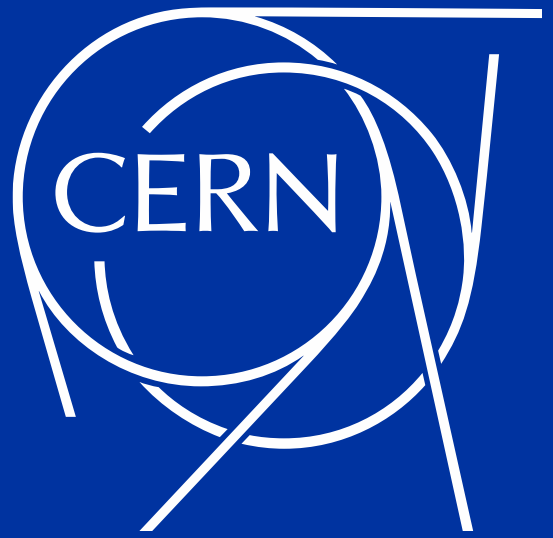
Microtests results - Less Jittery ops!





eos-filemd-convert

- **Tool** to convert/inspect file consistency metadata from one backend to another
  - LevelDBs have a lock file indicating some process has opened the file, so requires the corresponding FST for the filesystem to be stopped
- Also has a simple **inspect** mode to read fsck metadata for a given file
- In case the tool is being used for leveldb->attr conversions, it has to be made sure that FST started with attr as backend after
  - if not we'll miss the updates and FSTs commit new files to LevelDBs!
  - Already converted files are traversed, but not modified in FST startup
- Recommended to drop the .eosattrconverted touch file in case you want to force the FST to convert


CERN