
CERN

New approach of FST metadata storage

József Makai, CERN

As of today

- EOS storage servers has stored metadata so far in different (relational) databases
  - SQLite (Aquamarine)
  - levelDB (Citrine)
- Lot of maintenance operation on FST related to metadata databases
  - Detect and remove ghost entries
  - Start/shutdown the database, related error handling
  - Trim the database
  - …
- Lot of additional maintenance, resource utilization

New implementation

- Idea: store metadata of file as extended attribute attached to the physical file replica on disk
  - Base64 encoded, serialized protobuf metadata object
- Advantages:
  - No database maintenance, saving resources
  - Lot of related code can be completely thrown away, simplifies life
  - Doesn't have to execute queries to get metadata, only IO → less overhead

Compression of metadata

- This gave us the perfect opportunity to start compressing metadata
  - → compress → write
  - ← decompress ← read
- Metadata on storage servers can consume space up to ~10-100 GB
- Less IO in exchange for CPU on disk servers

ZStandard compression – part 1

- It can be trained with existing metadata for better compression → creates a dictionary
- Quite slow → relatively small training sets (~100k)

Size of dict.	Avg. ttc.	Avg. comp.	Best comp.	Worst comp.
100 KB	7 ms	0.528	0.123	0.838
400 KB	50 ms	0.479	0.102	0.822
600 KB	75 ms	0.459	0.101	0.822
1 MB	170 ms	0.445	0.101	0.871

ZStandard compression – part 2

- ZStandard is not thread safe, not feasible in concurrent environment
- It has a stateful context
- Protect the context with a mutex → one (de)compression at a time → not scalable
- We had to extend it with a wrapper for concurrency
- We need more contexts
  - thread_local context, one per thread
  - more contexts in concurrent stack → won the benchmark

Automatic conversion of metadata

- Supported for Citrine and levelDB metadata
- At FST startup, it tries to detect whether convert or not
  - It samples files from all mountpoints on FST
  - If >= 20% of files don’t have new metadata, it will initiate the conversion
  - Booting state until finished
- Uses a thread pool and all CPUs to parallelize
  - 5-10 minutes for EOSPPS FSTs (with ~50 million files)

Current state

- In nofstdb branch and not in master yet
- Also led to a new implementation of the fsck command
- Tested in CI, Docker image ready with the dictionary
- Some preliminary tests for EOSPPS conversion
- But it is coming soon!

Thank you! Questions?

József Makai, CERN