CERN Tape Archive: a distributed, reliable and scalable scheduling system

Eric Cano[1] Vladímir Bahyl[1] Cédric Caffy[1] Germán Cancio[1] Michael Davis[1] Oliver Keeble[1]
Viktor Kotlyar[2] Julien Leduc[1] Steven Murray[1]

[1]CERN, Geneva
[2]Institute for High Energy Physics, Protvino, Russia

CERN Tape Archive and EOS

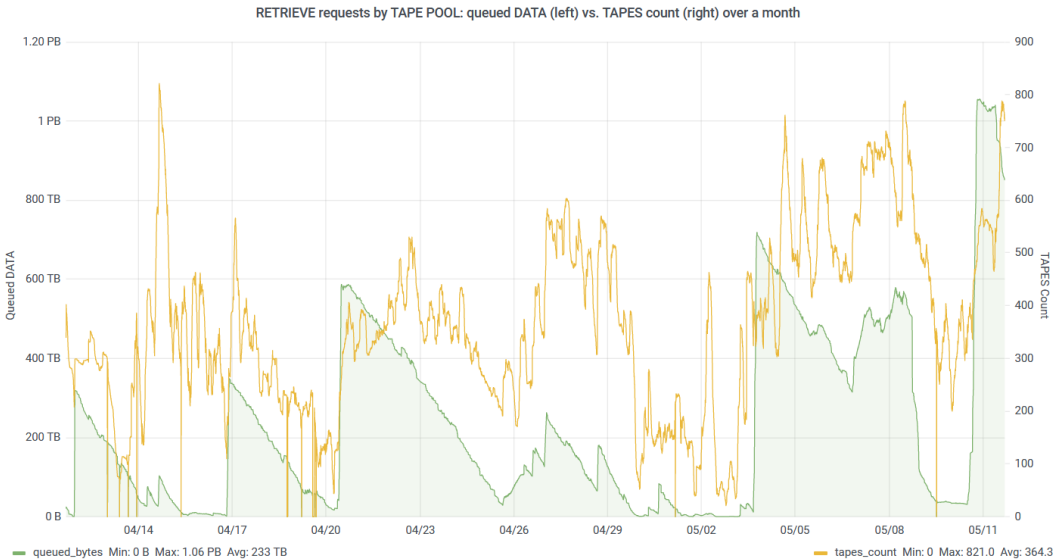
- CERN Tape Archive (CTA) is a tape backend for EOS and possibly dCache
- EOS CTA is a replacement for CASTOR instances
- CTA manages all aspects of the tape system
  - Physical location on tape: file catalogue
  - Data transfer: tape server
  - Transfer queuing: scheduling system
- All main 4 LHC experiments migrated from CASTOR to CTA
- Other experiments have or are being migrated

Issues and opportunities with CASTOR + EOS

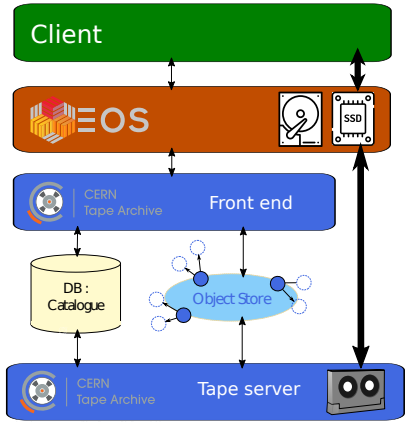
- EOS and CASTOR disk systems are redundant services
- CASTOR dependent on Oracle's PL/SQL at its core and hence vendor locked-in
- Design constrained by past database limitations that no longer exist
  - Limited number of connections to a database server
- Scattered logic over multiple databases and daemons in CASTOR
  - Multiple databases hold parts of the data
  - Multiple daemons take decisions in incremental steps lacking global view
  - Ex: Commit to write to a tape located in a library with no free drive
  - Communication channels between daemons costly to maintain and evolve

Tape queueing: more than FIFOs

- Tapes is the cheapest storage, with good bandwidth
  - 360 MB/s per drive
- But very high latency (1 min)
- Efficiency highly dependent on scheduling
- Transfers are accumulated until enough work warrants a mount
- Up to 10^6s of files are queued on 100s to 1000s of queues
- Mount scheduling decides which queue to process
- Then, within a mount, which files to process

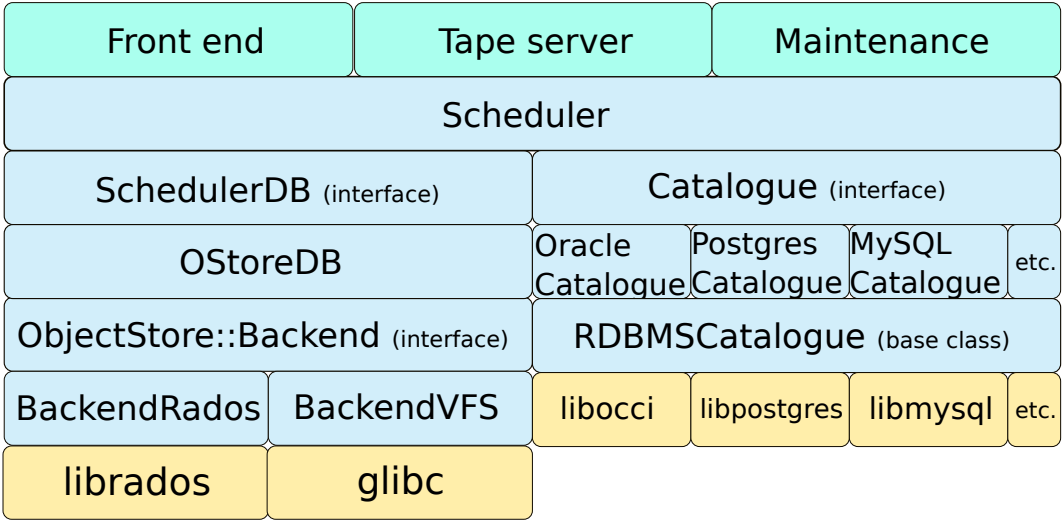

RETRIEVE requests by TAPE POOL: queued DATA (left) vs. TAPES count (right) over a month
1.20 PB
1 PB
800 TB
600 TB
400 TB
200 TB
0 B
Queued DATA
900
800
700
600
500
400
300
200
100
0
TAPES Count
04/14
04/17
04/20
04/23
04/26
04/29
05/02
05/05
05/08
05/11
queued_bytes Min: 0 B Max: 1.06 PB Avg: 233 TB
tapes_count Min: 0 Max: 821.0 Avg: 364.3

CTA architecture



HDD icon: https://commons.wikimedia.org/wiki/File:Hard-drive.svg
SSD icon: https://commons.wikimedia.org/wiki/File:Ssd.svg
Tape icon: https://commons.wikimedia.org/wiki/File:Tape_cinta_casette_backup.svg

- Direct access to shared metadata for all processes
- Two metadata storage systems
  - Database for persistent data: files and tapes catalogue
  - Object store for transient data: queueing information and drive statuses
  - Multiple backends for both
- Scheduler (C++ class): a single entry point shared by all processes of CTA
- Self-scheduling by drives: examines all queues, pick one to process
  - Filter possible mounts, prioritize, pick first


Front end
Tape server
Maintenance
Scheduler
SchedulerDB (interface)
Catalogue (interface)
OStoreDB
Oracle Catalogue
Postgres Catalogue
MySQL Catalogue
etc.
ObjectStore::Backend (interface)
RDBMSCatalogue (base class)
BackendRados
BackendVFS
libocci
libpostgres
libmysql
etc.
librados
glibc

Scheduler DB, aka Objectstore

- Tree of queues, requests, statuses for drives, tapes and processes
- Serialization based on Protocol Buffers
  - Only hard dependency in CTA
- Backends provide object level access
  - Lockable, atomically updated
  - Asynchronous access
  - Reliable
- Need multi-object coherence
  - With potential process crashes
- Scaling needed (few 100 Hz per queue)
  - 100s of drives
  - 1000s of queues
  - 10^6 files per queue

Reliability and scalability

- Algorithm achieving multi-objects operations safety
- Heartbeat based process crash recovery
- Scalability achieved with appropriate contention domains
  - Cutting structures in smaller parts where needed
  - Sharding queues for limited size of updates
- Lockless reads whenever appropriate
- Rate achieved by batching and asynchronicity
- Contention limited in practice to write access to queues
- Latency hiding mechanism added in front end (user visible)

Conclusion

- CTA successfully replaces the tape part of CASTOR after 20 years of service
- Provides a new scalable and efficient queueing and scheduling mechanism
- Flexible design allowing interfacing with other front-ends
- Designed for long-term with no vendor or technology lock-in