CTA ATLAS deployment

10/4/2019

IT/ST

Overview

- CTA migration strategy and open questions (Giuseppe and Michael)
- `eosctaatlas` deployment status, commissioning tests (Julien)
- Drive time efficiency and queueing latency (Germán)
- ATLAS activities & intra-VO fairsharing / storage classes (Eric)

CASTOR to CTA Migration

ATLAS tape files in CASTOR

- 84,062,944 files in group **zp**
- 86,689,714 files in file classes **atlas_raw**, **atlas_prod**, **atlas**, **atlas_user** (and 7 others like ***atlas*** with under 2m files each)
- Several thousand files with no tape copy (zero-length files; files in **atlas_no_tape** file class)

CASTOR Namespace

- `/castor/cern.ch/grid/atlas` (70,716,745 files)
- `/castor/cern.ch/atlas` (5,805,960 files)
- `/castor/cern.ch/user` (10,209,365 files)

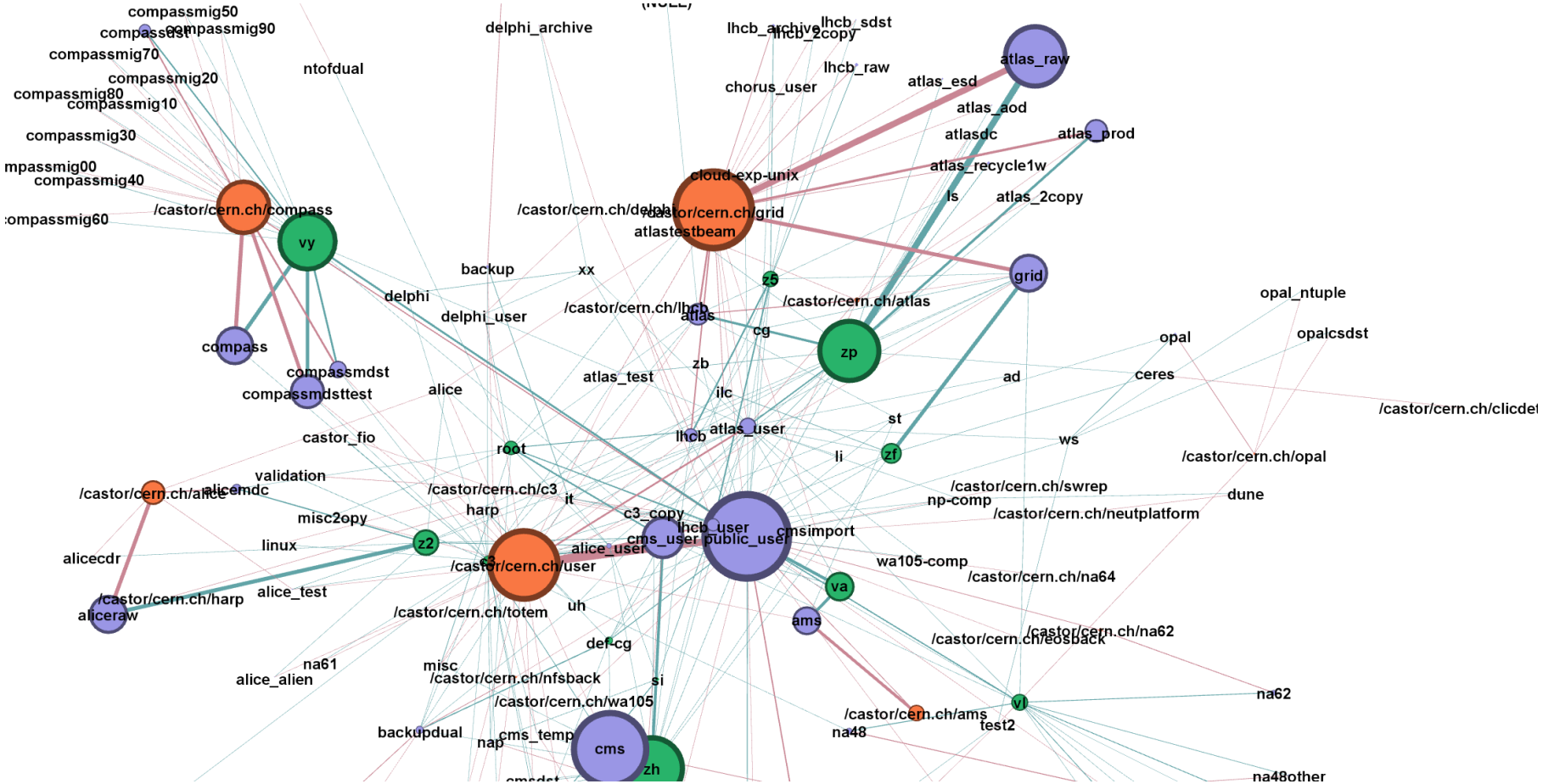
CASTOR Access Control

- Directories have POSIX permission bits (95% are **755** or **775**. 5% split over 50 combinations)
- 17% of directories also have Access Control Lists. Around 9,000 combinations.

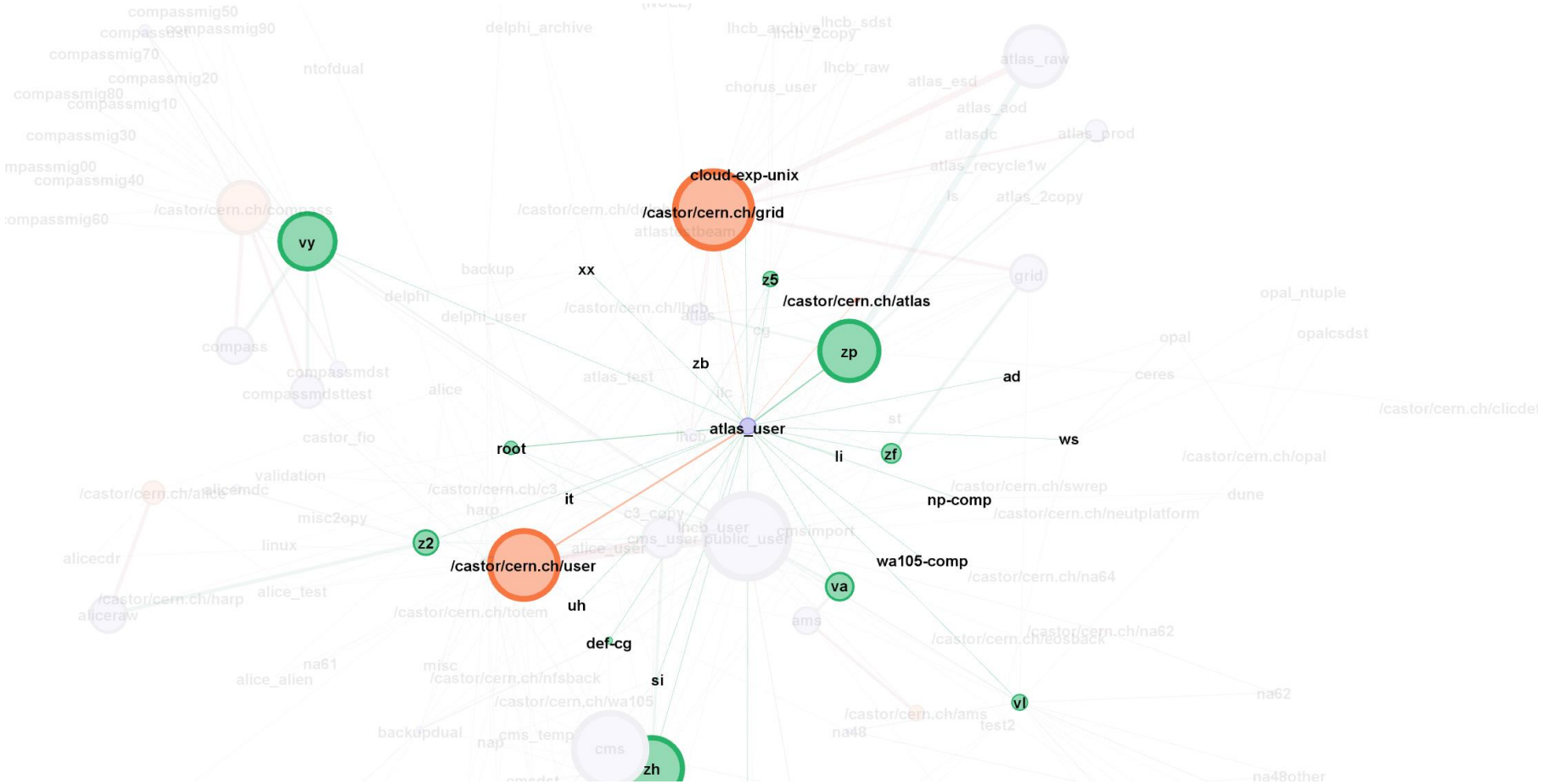
CASTOR to CTA Migration

- CASTOR has a single namespace. CTA will partition the namespace into five instances (one per LHC experiment and one for PUBLIC).
- How to determine which instance a file belongs to? There are three hints:
  - File class
  - Directory branch in the namespace
  - Group ID
- The minimum granularity for migration is a single cassette

CASTOR to CTA Migration



CASTOR to CTA Migration



Migration Questions (1)

Experiment and User file classes

- **atlas_user** will be migrated to the ATLAS instance
  - User data is mixed with legacy production data on the same tapes

We propose to NOT migrate the following metadata:

- Zero-length files, files with no tape copy, deleted files. (This metadata will remain accessible in the CASTOR namespace).
- Group IDs for individual files/directories. After the migration, all will be set to **zp**.
- POSIX permissions and Access Control Lists. A new set of permissions will be created in EOS+CTA according to ATLAS use cases.

Migration Questions (2)

Where should migrated files appear in the EOS+CTA namespace?

- We propose to have the new namespace under `/eos/cta/atlas`
- Option 1: Keep migrated data and new data separate
  - `/castor/cern.ch/grid/atlas/` → `/eos/cta/castor/grid/atlas/`
  - New data under `/eos/cta/atlas/`
- Option 2: Keep migrated data and new data in the same branch
  - `/castor/cern.ch/grid/atlas/` → `/eos/cta/atlas/`
  - New data under `/eos/cta/atlas/`
- Same question for `/castor/cern.ch/user/`

Migration Milestones

Preparation

- Agree how files (*i.e.*, tapes) will be partitioned between EOSCTA ATLAS and EOSCTA PUBLIC_USER
- Agree on access use cases : users, groups and permissions
- Migrate metadata to test instance (files remain accessible only from CASTOR)

Live Migration

- Select files to be migrated; disable the tapes in CASTOR
  - Subsequent metadata operations on these files (delete, rename) are strongly discouraged!
- Copy metadata to intermediate table in CTA database (DBLINK)
- Inject directory metadata into EOS namespace
- Inject file metadata into EOS namespace
- Inject tape file metadata into CTA catalogue
- Enable tapes in CTA

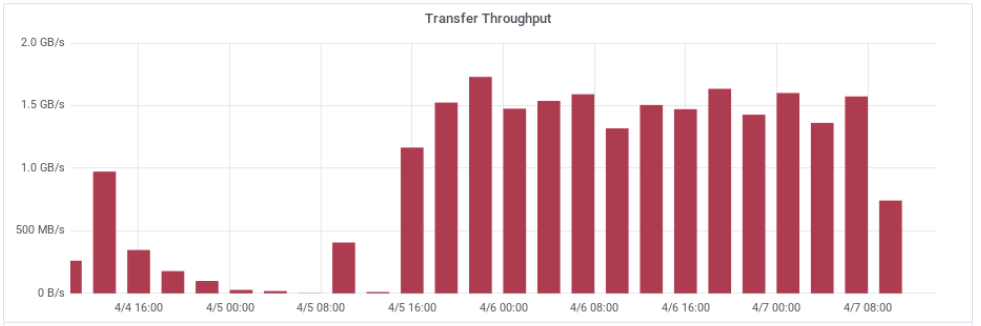
Disaster Recovery/Rollback

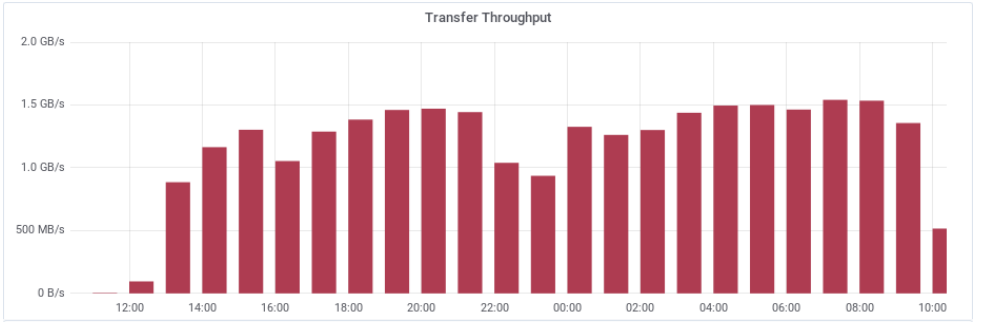
- CTA will be prohibited from writing to tapes imported from CASTOR
- To return a tape to CASTOR, disable the tape in the CTA catalogue and re-enable the tape in CASTOR

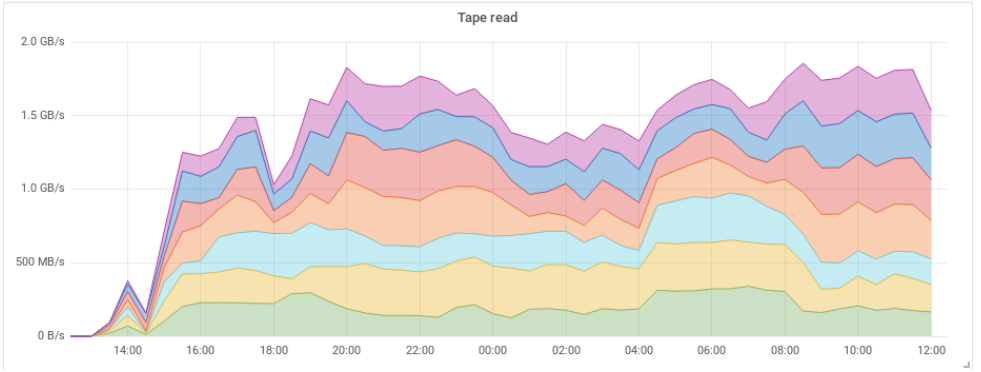
CTAATLASPPS transfers

- First ATLAS data archival to CTA with SSDs
  - 16TB of SSD buffer, 7 tape drives
  - 250TB of data were written to tapes
  - Average throughput 1.5 GB/s
  - Transfer efficiency of 20% due to missing free space feedback (planned FTS feature)

- First ATLAS data retrieval from CTA with SSDs
  - 250TB of data are currently being retrieved
  - Average throughput 1.5 GB/s
  - Transfer efficiency of 96%

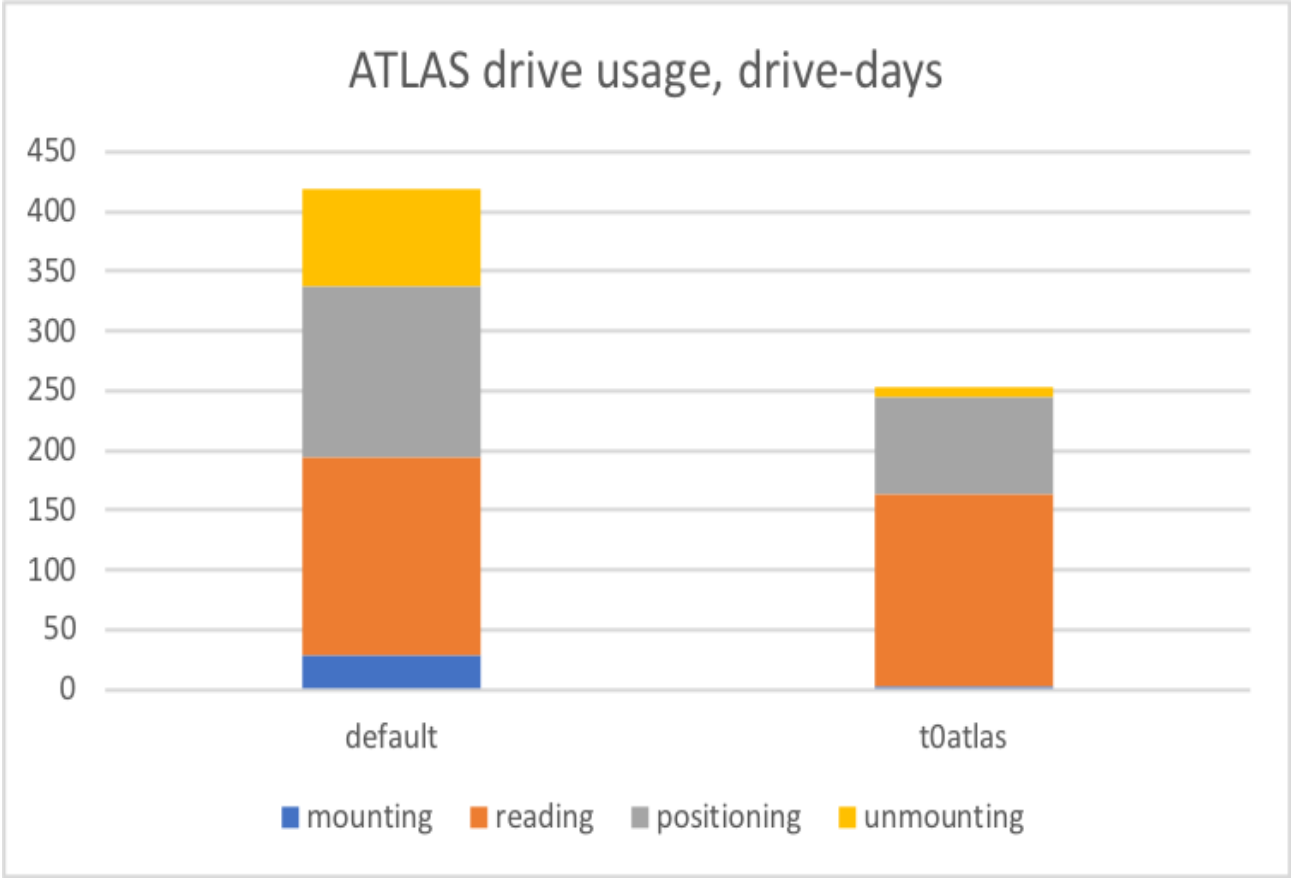






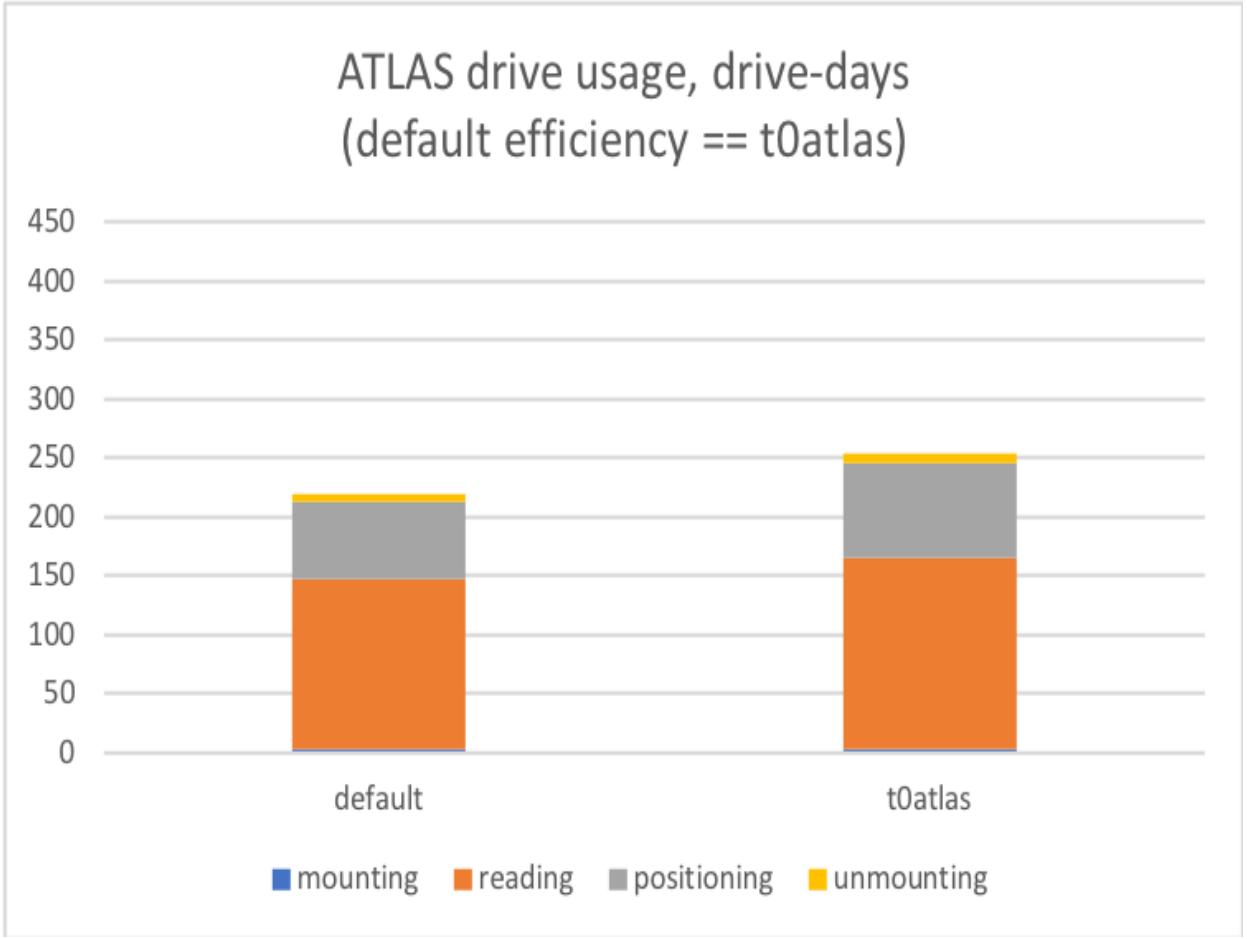
ATLAS and overall drive time efficiency

- ~2/3 of ATLAS drive utilisation adsorbed by “default” - despite only 45% of data
  - Large impact of mount, unmounting and positioning (1st file)
- With dataset-level reading, “default” drive usage would be ~50% less…
- … liberating drive resources (less queueing/latency, more parallel reading)



ATLAS and overall drive time efficiency

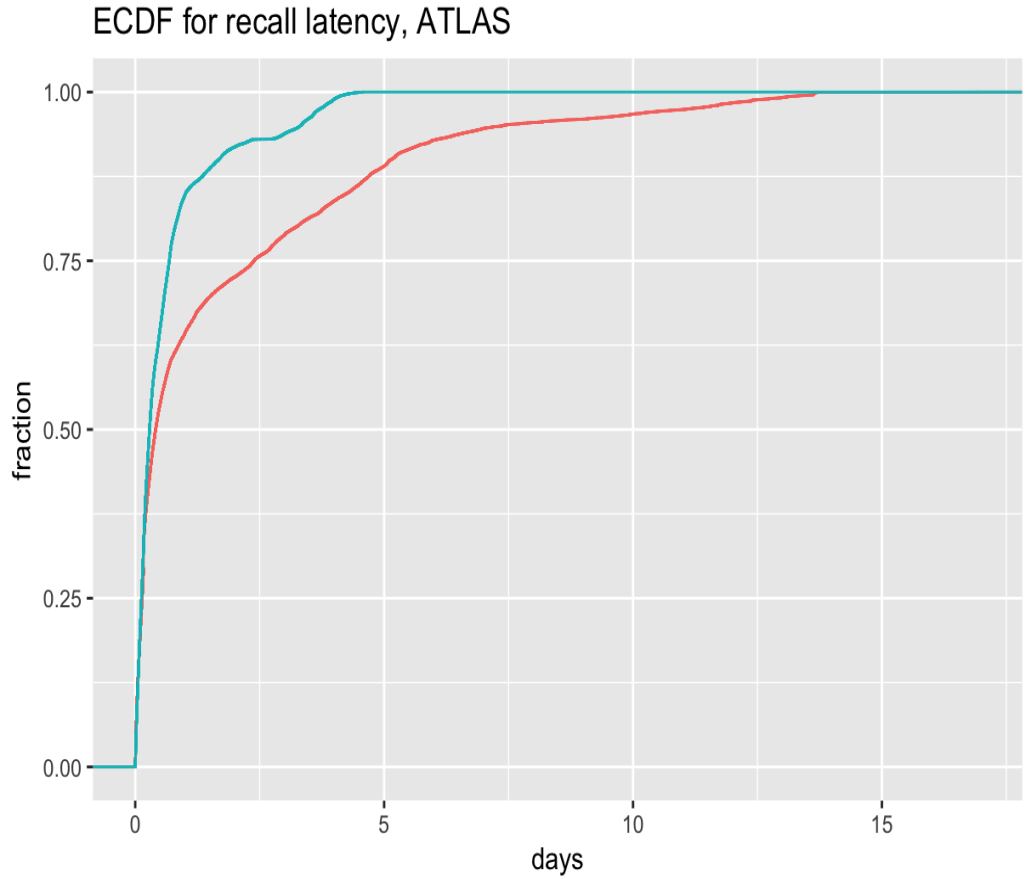
- ~2/3 of ATLAS drive utilisation adsorbed by “default” - despite only 45% of data
  - Large impact of mount, unmounting and positioning (1st file)
- With dataset-level reading, “default” drive usage would be ~50% less…
- … liberating drive resources (less queueing/latency, more parallel reading)

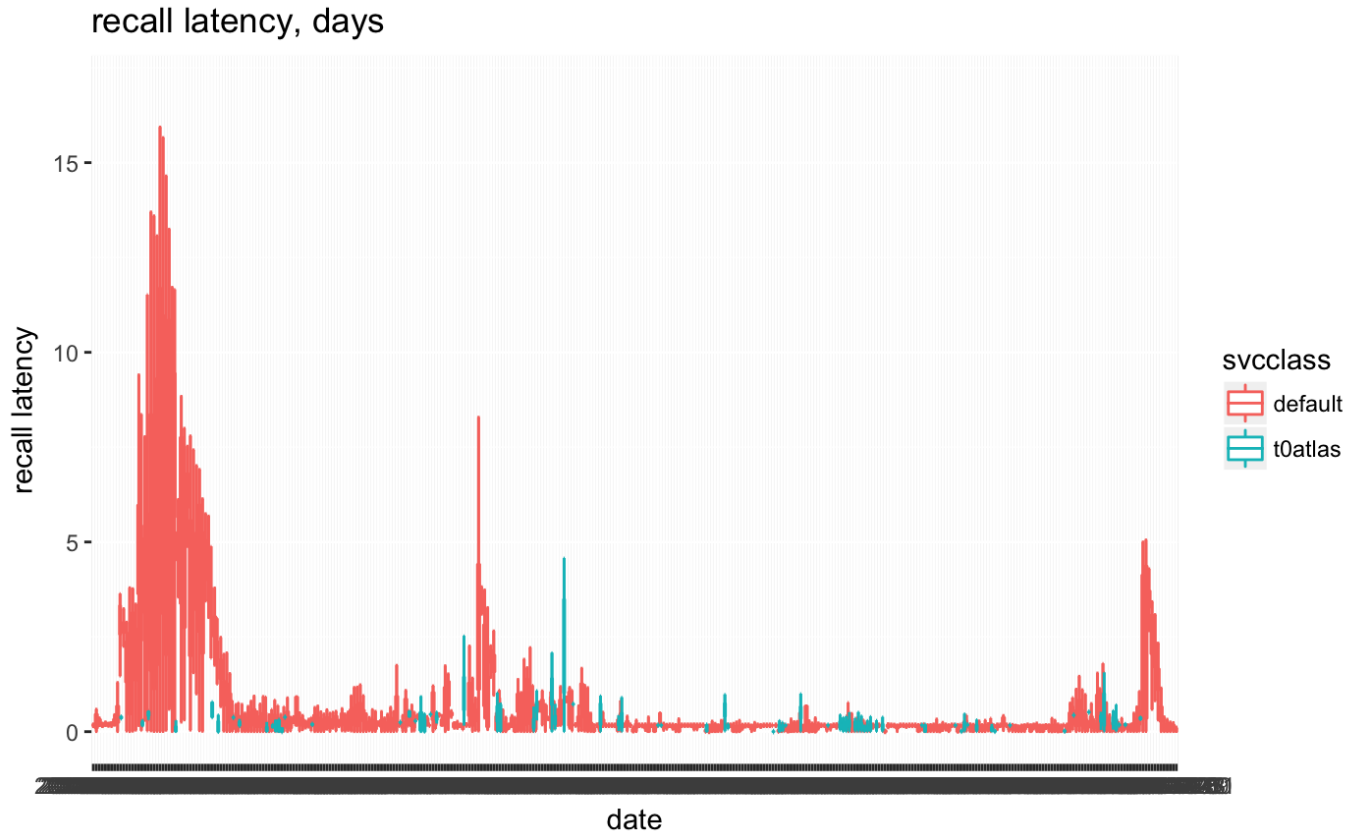


ATLAS 2018 queueing latency

Latency (time to last bit of file):

- default: median 0.41 d; mean 1.7 d
- t0atlas: median 0.29 d; mean 0.64 d





ATLAS activities / intra-VO fairsharing / storage classes

- FTS will propagate activity and CTA will honour it
  - Activities configuration in FTS mirrored in CTA (map of tags to integer weights)
  - Activity tag passed through Xrootd and EOS by FTS
  - Mounts arbitrated between activities (weighted fair share)
- CTA will then arbiter retrives using either FIFO or Bandwidth criteria
  - All FIFO for Atlas, mixed case possible.
- Cap of “parallel writes” for a given tape pool
  - Parallel writing: how many tape do we simultaneously open for writing at the same time?
  - But... creates an upper bound on the migration bandwidth (caps the number of parallel mounts)
  - Tape dedicated to either repack or new data within a write mount.
- Fail retrieves with more information, and faster (new possible idea)
  - Fail requests earlier (and with more detail) when the data will not come in the foreseeable future
    - Example: Tape sent to repair
  - Discriminate with transient problems
    - Example: disk buffer full

Storage class:
- Number of copies (1 in our case)
- Copy number to tape pool map (1 in our case)

Tape pool:
- Owning VO
- Max number of partial tapes (parallel writes)
- Encryption

Mount Policy
- Max archive and retrieve drives
- Retrieve prioritisation: <Bandwidth|Latency> (to be added)

Activities (to be added)
- Tag
- Weight

CTA and tape read efficiency

- The CTA project is looking at improving resource efficiency from several angles
  - optimising tape and drive scheduling (integrated in CTA sw)
  - read access ordering on LTO ("CERN RAO") – WIP
  - minimising disk contention and capacity waste (Julien's SSD disk layer)
  - Others
    - larger file sizes – requires collaboration with experiments
    - efficient pre-staging (complete data sets)
    - collocation hints (Archival WG future output) for increasing contiguous file access

Why do we split data across tapes?

Operational reasons that require splitting of input streams during writes:

- ensure aggregated performance and time-to-tape latency SLA's in writing (each Run-3 tape drive will only sustain 0.3-0.5 GB/s) (cf Julien's slides at Rucio workshop)
- non-availability of tapes (eg. under repair, stuck in a drive)
- long library queueing wait times (busy drives, robotics)
- library (segment) downtime or maintenance
- potential impact of "holding back to collocate" on CTA buffer (and pit/T0 buffer)

- Tape technology will be growing faster in capacity (~20% CAGR) than in throughput (10-15% CAGR)
- Future evolution may require us to consider striped tape writing such as RAIT (cf as done in HPSS)

How does this splitting affect performance?

- **Performance:**

$$f(collocation, average\ file\ size, number\ of\ available\ tape\ drives)$$

- Collocation refers to both request and tape collocation
  - How related are files being processed? How far away are they across tapes?
- What counts is the resulting latency and throughput
  - For ATLAS: Latency to last byte of each dataset
- Queue fairsharing and FIFO ordering will address ATLAS latency issues
  - cf Eric's slides
- Controlled tape multiplexing increases r/w throughput and latency
  - Collocation *within* a multiplexed tape helps as well, as a second-order optimisation

Example

- Assumption: ATLAS reading out, then processing complete (large) datasets
- Dataset size: 10TB; nominal drive speed: 350MB/s; effective speed factor: 0.7 (0.3 for positioning) -> 245MB/s
- Case a) fully collocated writing -> single tape @ factor 1

Fully collocated:	Read time (s)	Read speedup factor	Effective per-drive speed factor
1 drive	2857.142857	1	1
2 drives	2857.142857		N/A

- Case b) multiplexed writing -> N tapes in parallel @ factor 0.7

Multiplexed:	Read time (s)	Read speedup factor	Effective per-drive speed factor
1 drive	4081.632653	0.7	0.7
2 drives:	2040.816327	1.4	0.7
3 drives:	1360.544218	2.1	0.7
5 drives:	816.3265306	3.5	0.7

- → Multiplexing enables substantial latency and performance gains
- NB: Tape drive cost is not part of ATLAS pledge but CERN/IT; operational overheads are borne by CERN/IT in any case (0.3 tape drive overhead, EOS disk overhead due to redundancy, switch/router network overheads, etc)