PostgresSchedDB overview and status update

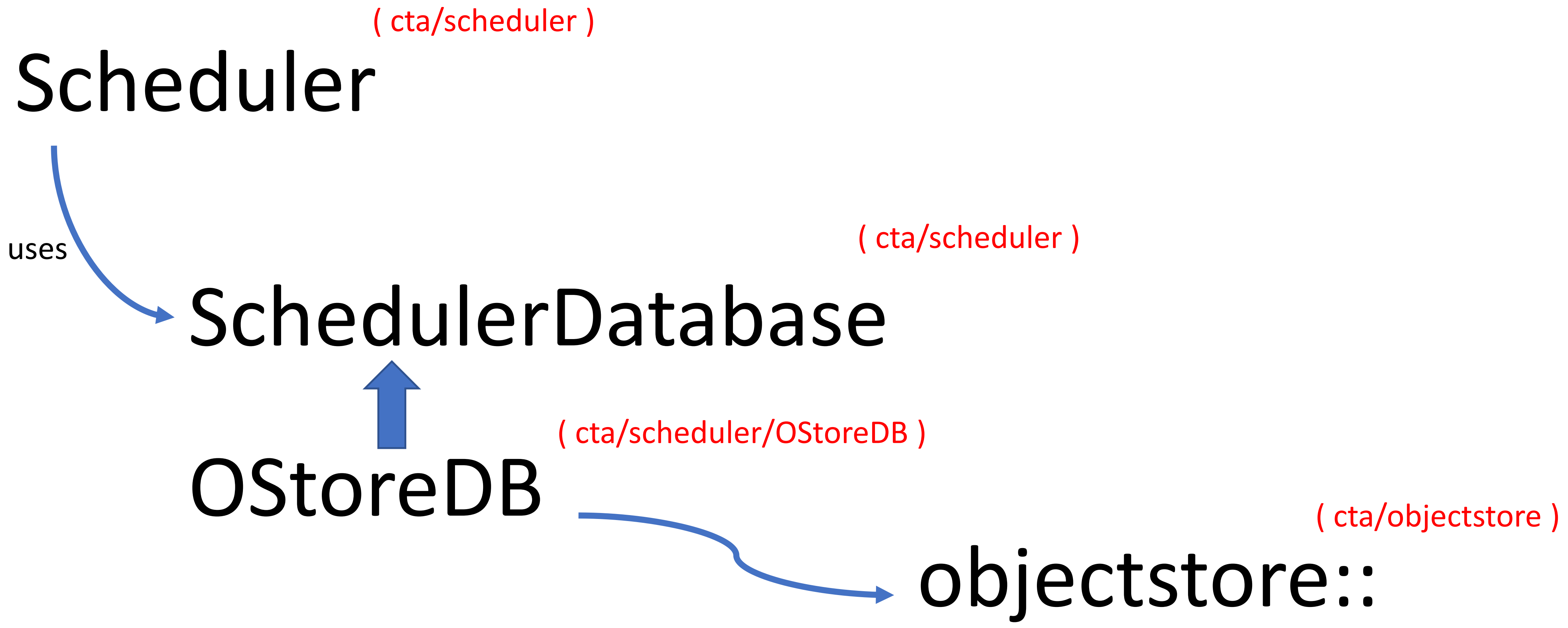
30 Mar 23

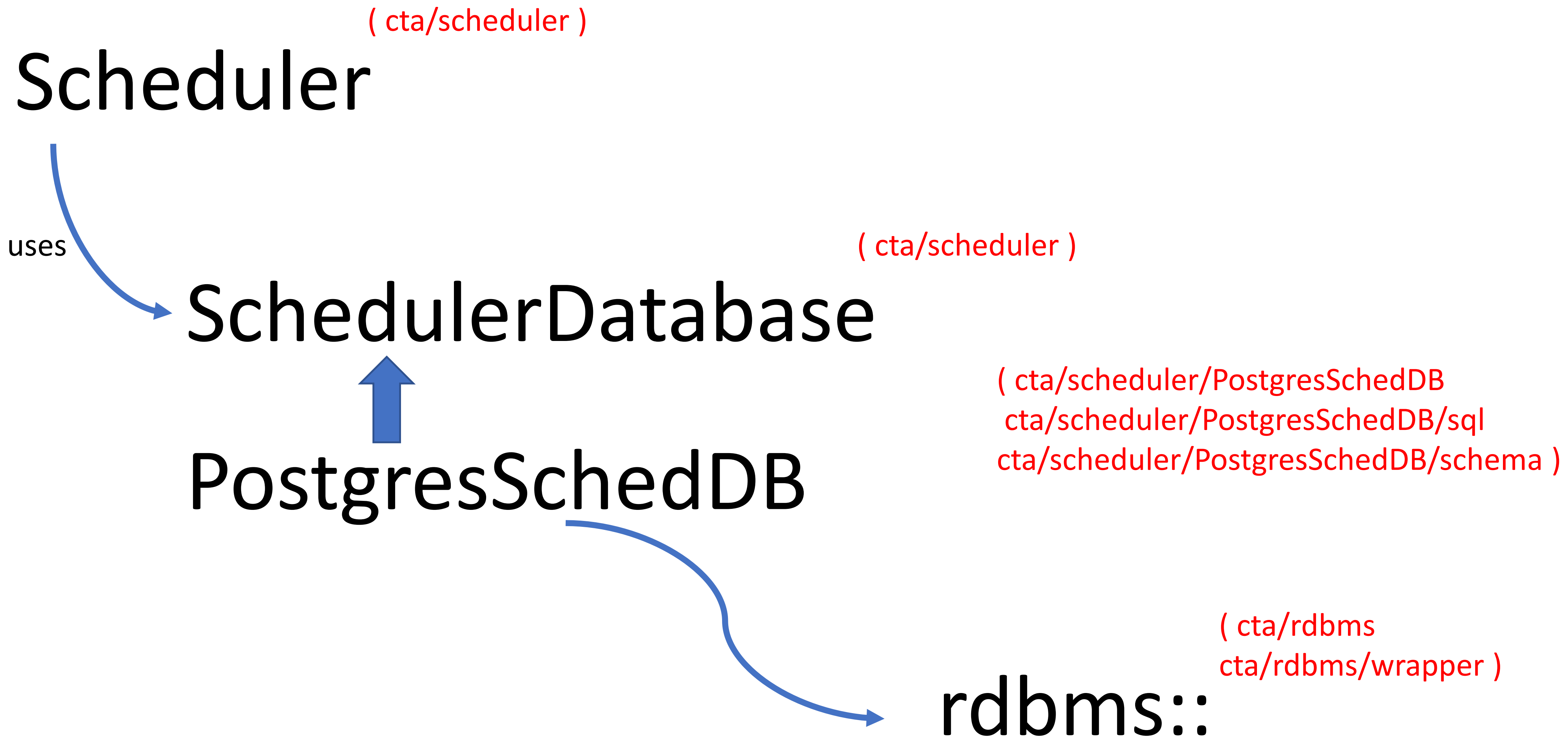
CTA developers meeting

David Smith

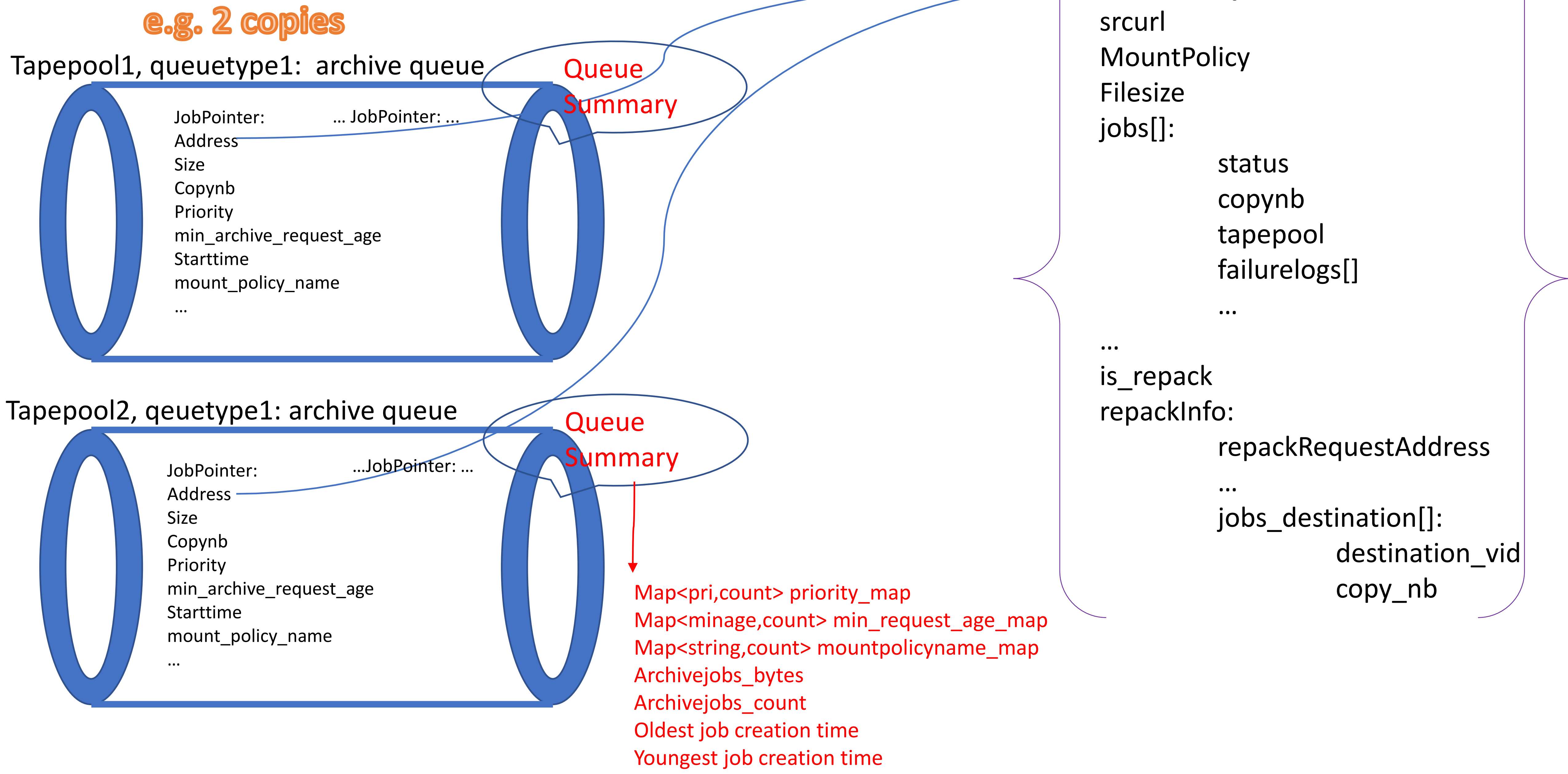
Introduction

- Quick note on Names: Scheduler, SchedulerDatabase, OStoreDB, objectstore are C++ names (class or namespace) and also reflected in subdirectory names in the CTA source tree
- This development concerns (mostly) the storage and retrieval of the data on which the Scheduler works: data contain information about requests and their status and let the Scheduler drive the request life cycle
- Requests: Archive, Retrieve and Repack.
- Lifecycles: Enqueue (typically at frontend), allow cta-taped to perodically choose a tape to mount to fullfill a set of Archive or Retrieve (and thus Repack) requests. Once a tape is mounted, must allow cta-taped to read or write the appropreate files to tape.


(cta/scheduler)
Scheduler
uses
(cta/scheduler)
SchedulerDatabase
(cta/scheduler/OStoreDB)
OStoreDB
(cta/objectstore)
objectstore::


(cta/scheduler)
Scheduler
uses
(cta/scheduler)
SchedulerDatabase
(cta/scheduler/PostgresSchedDB
cta/scheduler/PostgresSchedDB/sql
cta/scheduler/PostgresSchedDB/schema)
PostgresSchedDB
(cta/rdbms
cta/rdbms/wrapper)
rdbms::

Objectstore (brief!): Archive



Postgres table: Archive_Job_Queue

Flatten the request + job information into repeated rows, one row per job (except for “failure logs”). e.g. for a 2 job archive request as this:

srcurl	filesize	Status (1)	Copynb=1	Tapepool (1)	Min_archive_req_age	…	Faillogs_PB (1)	Is_repack	destinion_VID (1)
srcurl	filesize	Status (2)	Copynb=2	Tapepool (2)	Min_archive_req_age		Faillogs_PB (2)	Is_repack	destinion_VID (2)

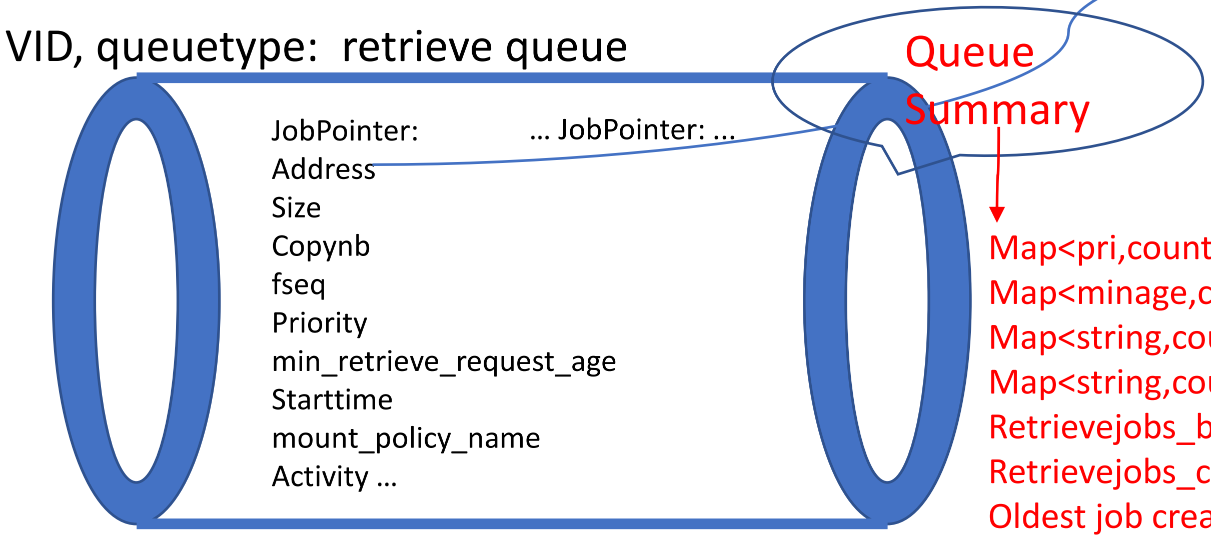
(39 columns in current schema)

Repeated request level information rather than having a separate request & job tables. Flattern the repackInfo data into the row as well.

Have a column with ProtoBuf content: the failurelogs / reportfailurelogs which contain failure information for retired jobs.

Objectstore (brief!): Retrieve

e.g. for file with 2 tape copies



Map<pri,count> priority_map
Map<minage,count> min_request_age_map
Map<string,count> mountpolicyname_map
Map<string,count> activity_map
Retrievejobs_bytes
Retrievejobs_count
Oldest job creation time
Youngest job creation time

RetrieveRequest:

```
Scheduler_request
        dstURL
        …
MountPolicy:
        name
        min_retrieve_req_age
        retrieve_priority …
activity
ArchiveFile:
        archivefileId
        TapeFile tf[]:
                copyNb
                vid
                fseq
                …
Active_copy_nb
RetrieveJob jobs[]:
        copyNb
        maxTotalRetries
        TotalRetries
        …
        failurelogs[]
        isFailed

isVerifyOnly
isFailed
isRepack
RetrieveReqRepackInfo:
        …
```

Job queued once, although there may be multiple tape copies. Queue of JobPointers kept in fSeq order.

Tape copies to be considered are provided by the Scheduler to SchedulerDatabase; this may be a subset of those in the Catalogue.

The queue, and hence VID (tape) is determined within the SchedulerDatabase (or lower) layer any time the job needs to be (re)-queued; i.e. using the *objectstore::Helpers* methods.

Postgres table: Retrieve_Job_Queue

Decided to only queue one row for a Retrieve job, but need to have enough information to (re)-try at any VID.
Flattern as much as possible, with some columns reflecting the current "active" copy.

Status (active_copynb)	Vid (active_copynb)	Min_retrieve_req_age	Active_CopyNb	…				RetrieveJob_PB	RepackInfo_PB

Combine TapeFiles and Jobs arrays into a single repeated protobuf, since there is always a TapeFile <-> Job relationship.

RepackIfno is used in the repack workflow, to buid an ArchiveRequest once the RetrieveRequest is done.

The RepackInfo contains repated elements, e.g. archive_routes and copy_nbs_to_rearchive that depend on number of new Archives to be created. The RetrieveJob contains entries corresponging to the number of possible soruce tape copies; so RetrieveJobs and RepackInfo were kept separate.

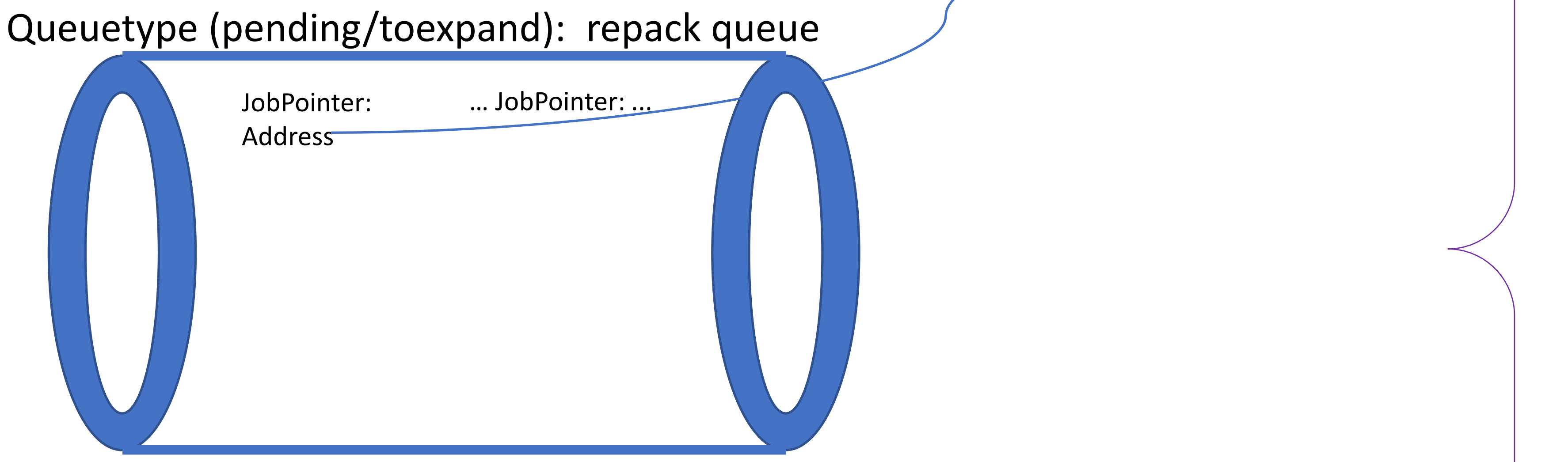
Postgres views: Archive_Job_Summary & Retrieve_Job_Summary

To provide the equivalent of the per Queue summary information a Query View is created. The query groups by tapepool or VID, status, mountpolicy_name, returning count, and min/max(start_time), sum(bytes).

QueueTypes

Not imagined to specifically preserve concept of queuetype in the postgresDB; this is implied by state or state + isRepack. Planned to add FAILED_ tables (x 3 for archive/retreieve/repack) for requests in terminal state.

Objectstore (brief!): Repack



RepackRequest:
Vid
Buffer_url
Status
Add_copies_mode
Move_move
Totoal_files_to_retrieve
…
Failed_to_archive_bytes
lastExpandedFseq
…
Is_expand_finished
Is_expand_started
Mount_policy
…
Subrequests[]
Destination_infos[]

Postgres table: Repack_Job_Queue

Request already quite flat: Indicded two protobuf columns for subrequests[] and destination_infos[].

Ownership

- Ownership
  - Plan to maintain an ownership concept: one column for owner in archive/retrieve/repack. PostgresDB peridically update an owner-table with a heartbeat. => still a garbage collection collection;
  - select for update from owner-table where “heartbeat too old”;
  - Remove ownership/adjust matching archive/retrieve/repack reqs.
  - Drop old owner row
- Would be nice to not have this but this is what I’m working towards..
  - Considered keeping row locks while jobs active, but long lived db transactions likely to be problematic
- Request IDs
  - With objectstore, IDs (“addresses”) are computed before creation. Natural approach for DB is to have a primary key for a table as a sequence, which thus is discovered after insertion.

Status

- Much to do, but several steps done
  - Schema more or less complete; but surely subject to some change
  - All written code compiling, but not functional
  - Request/job objects written, with insert of new requests and retreieve from database and serialisation/deserialisation of the protobuf columns.
  - Initial creation/queuing functions added (in the PostgresSchedDB class) .
  - fetchMountInfo fetching possible Archives. (Missing retrieve at moment!)
  - createMount for Retrieve & Archive present.
  - Postgresscheddb::Helpers equivalent added to select best VID for Retrieve.
  - Initial part of expansion for Repack (creation of RetrieveRequests) added.

Expected next steps (of development)

- Add (untested) code for obvious missing functions
  - fetchMountInfo for Retrieve, and creation of archive requests for the repack workflow (what is called the Transfation in the objectstore).
  - Ensure appropreate use passing of connection pool vs transaction
  - Adopt ID (vs address) style indxexing of the rows (it's currently a bit mixed).
  - Reporting (maybe success first)
  - Then move to a stage of running small workflows with intense debugging of the methods step by step.
    - Also run tests (Much of SchedulerTest.cpp tests should eventually pass, have prepared a "GenericSchedulerTest" which excludes a few of the current tests which directly do actions on the objectstore).