Accès aux données - Qserv

D. Boutigny, S. Elles
LAPP/CNRS/IN2P3, Annecy

LSST France - APC - Novembre 2018

Outline

⇨ Qserv Database for the LSTT Stack software

from LSST datasets/catalogs to the Qserv database

From LSST datasets/catalogs to the Qserv database

Qserv is developed at SLAC + IPAC
Design optimized for astronomical queries (parallel distributed SQL database)

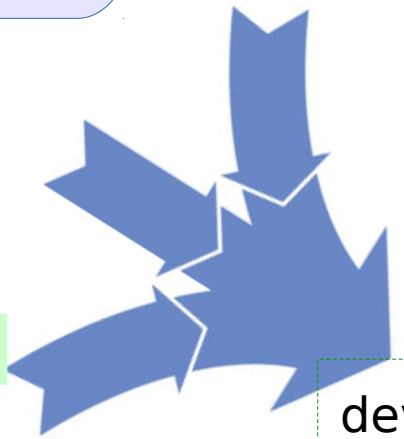
Starting point :

❶ **SQL DB structure**
defined in collaboration with the Stack developers & physicists

❷ **Qserv DB access tools**
developed by **Qserv** team
with the help of LPC Clermont Fabrice Jammes

❸ **Stack - cluster package**
Stack catalog browser - DRPtools -
developed by N.Chotard

❹ **CC-IN2P3 computing infrastructure**
Qserv cluster based on Kubernetes
with the help of F. Jammes, F. Wernli and F. Hernandez



Target

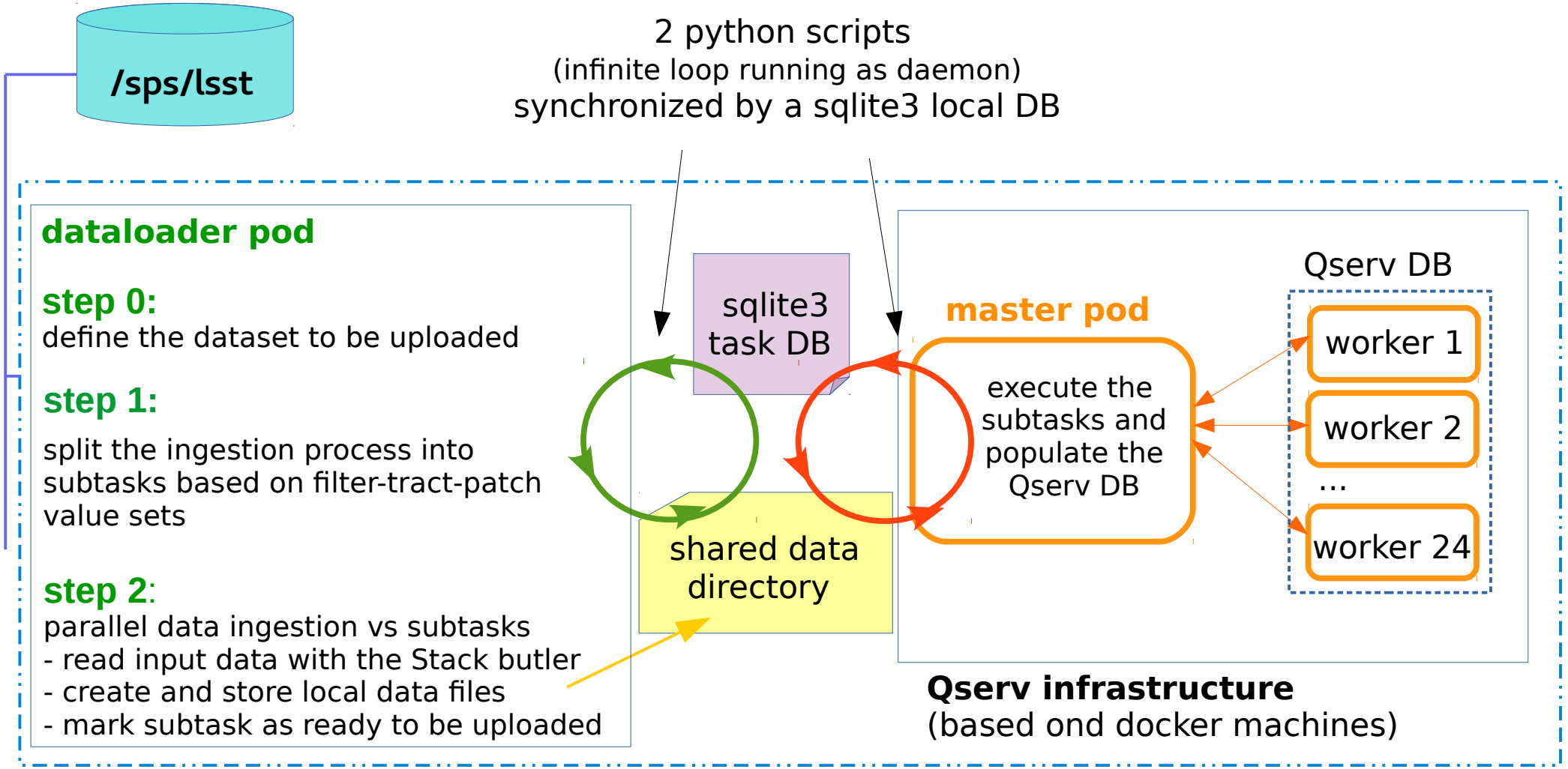
develop a tool to ingest the LSST stack datasets/catalogs into a Qserv DB

Guidelines
- asynchronous process (parallelisation)
- ensure data integrity
- deals properly with the error-recovery

Data ingestion

a prototype is available

Kubernetes Qserv infrastructure
- 1 master + 24 workers
- + a dataloader pod that coordinates the data ingestion



Data ingestion - current status

- Qserv data ingestion process based on Kubernetes is operationnal

  DB structure is configurable through yaml fille
  - DB table definition, parameters to be uploaded, …
  - Director table (that defines the way the sky is mapped into chunks)

  Runs in parallel mode to avoid pods to be heavy loaded

- No performance test made yet (load test, timing, …)

- Script/library to access the Qserv DB is available (from the CC interactive machines)

  Requests are built following the mysql formalism

```
# Describe the structure of a given table
python qserv_test_query.py --db qservTest_case120_qserv --request "describe deepCoadd_meas;"

# Filter defined in deepCoadd_meas
python qserv_test_query.py --db qservTest_case120_qserv --request "select distinct filter,tract,patch from deepCoadd_meas;"

# Number of entries in a given table
python qserv_test_query.py --db qservTest_case120_qserv --request "select count(*) from deepCoadd_meas;"
```

Current issues :

- Kubernetes seems quite sensitive to memory overload, pods can be killed without warning : using the cvmfs software releases is currently a problem
  ⇒ difficult to debug because the crashes are not reproducible

  in progress

- Qserv HTTP requests response size is underestimated, especially when the number of parameters defined in a table grows (Run.1.1 => 845 parameters)
  ⇒ to be solved soon - Qserv team

  problem solved by A. Salnikov

Data ingestion - current status

- Script to acces the Qserv DB is available (from the CC interactive machines)

  MACSJ2243.3-0935 dataset is available in the DB
  ⇒ to be used to validate the ingestion process

  The Qserv DB query library uses the DRPtool & Qservi tools developed by N. Chotard to access data through the Stack butler

  ⇒ straightforward to switch from analysis using the butler to Qserv queries

Next steps :

- Switch to a new CC-IN2P3 Kubernetes infrastructure - provided by F. Jammes
  ⇒ operation planned for Nov/Dec 2018
- Upload larger data catalogs to create a Qserv analysis testing DB.
  Next step : upload a Run1.2 dataset (see “current issues” in previous slide)

first trial with a reduced number of parameters