Base de données LSST/Qserv

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

LSST France - CPPM - Janvier 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**

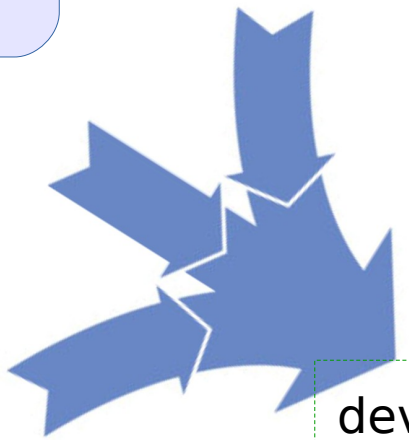
developed by **Qserv** team - collaboration & support from LPC Clermont (F. Jammes)

❸ **Stack - cluster package**

Stack catalog browser developed by N.Chotard

❹ **CC-IN2P3 computing infrastructure**

Openstack Virtual machines



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

From LSST datasets/catalogs to the Qserv database

❶ **SQL DB structure**

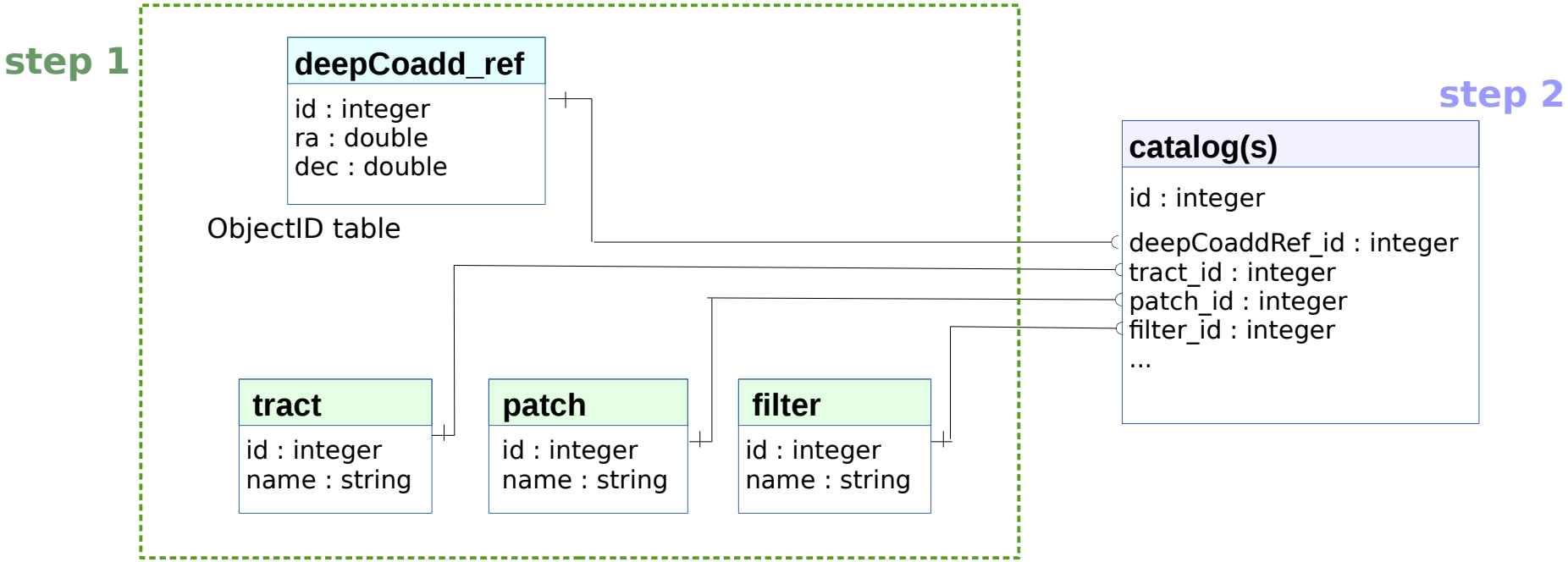
defined in collaboration with the LSST stack developers & physicists

⇨ split datasets/catalogs over several tables linked through a SQL foreign key mechanism

+ unicity constraints

- tract, patch, filter parameters stored in independent SQL tables — parameter name
- stack objectId (+object coord.) stored in an independent table — objectId
- catalogs stored one per table with reference to the tract, patch, filter and objectId tables — (tract,patch,filter,objectId)

↳ loading the Stack catalog is a 2-step process



From LSST datasets/catalogs to the Qserv database

SQL DB structure

defined in collaboration with the LSST stack developers & physicists

↪ the DB structure defined by Stack is configurable through python script

db catalog data configuration

```
DataModel={
   "deepCoadd_forced_src": {"autoincrement": True,
                        "cat_idKey": "id",
                        "primarykey": "deepCoadd_forced_srcId",
                        "dependencies": (["deepCoadd_ref", "deepCoadd_ref_fkId"],["filter", "filter_fkId"]
                                                ,["patch", "patch_fkId"["tract", "tract_fkId"],),
                        "constraint":{"unique":('tract_fkId','filter_fkId','patch_fkId','deepCoadd_ref_fkId')},
                        "properties" : ["partitioned"],
                        },
   ,
   "filter":{ "autoincrement": True,
              "cat_idKey": "filter",
              "cat_params" :("filter"),
              "primarykey": "filterId",
              "dependencies": None,
              "constraint":{"unique":('filter',)},
            }
```

db filter table configuration

....

```
  "deepCoadd_ref":{  "autoincrement": True,
                              "cat_idKey": "id",
                              "cat_params":("id","coord_ra","coord_dec"),
                              "primarykey": "deepcoadd_refId",
                              "dependencies":None,
                              "constraint":{"unique":('id',)},
                              "properties" : ["director"],
                              }
```

db objectId table configuration

...

From LSST datasets/catalogs to the Qserv database

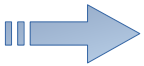
❷ Qserv DB access tools

developed by **Qserv** team - collaboration & support from LPC Clermont (F. Jammes)

Qserv DB tools available

⇨ deployment of a set of master/worker Qserv docker machines
python script to populate a Qserv DB
(based on cvs -comma separated values- files)

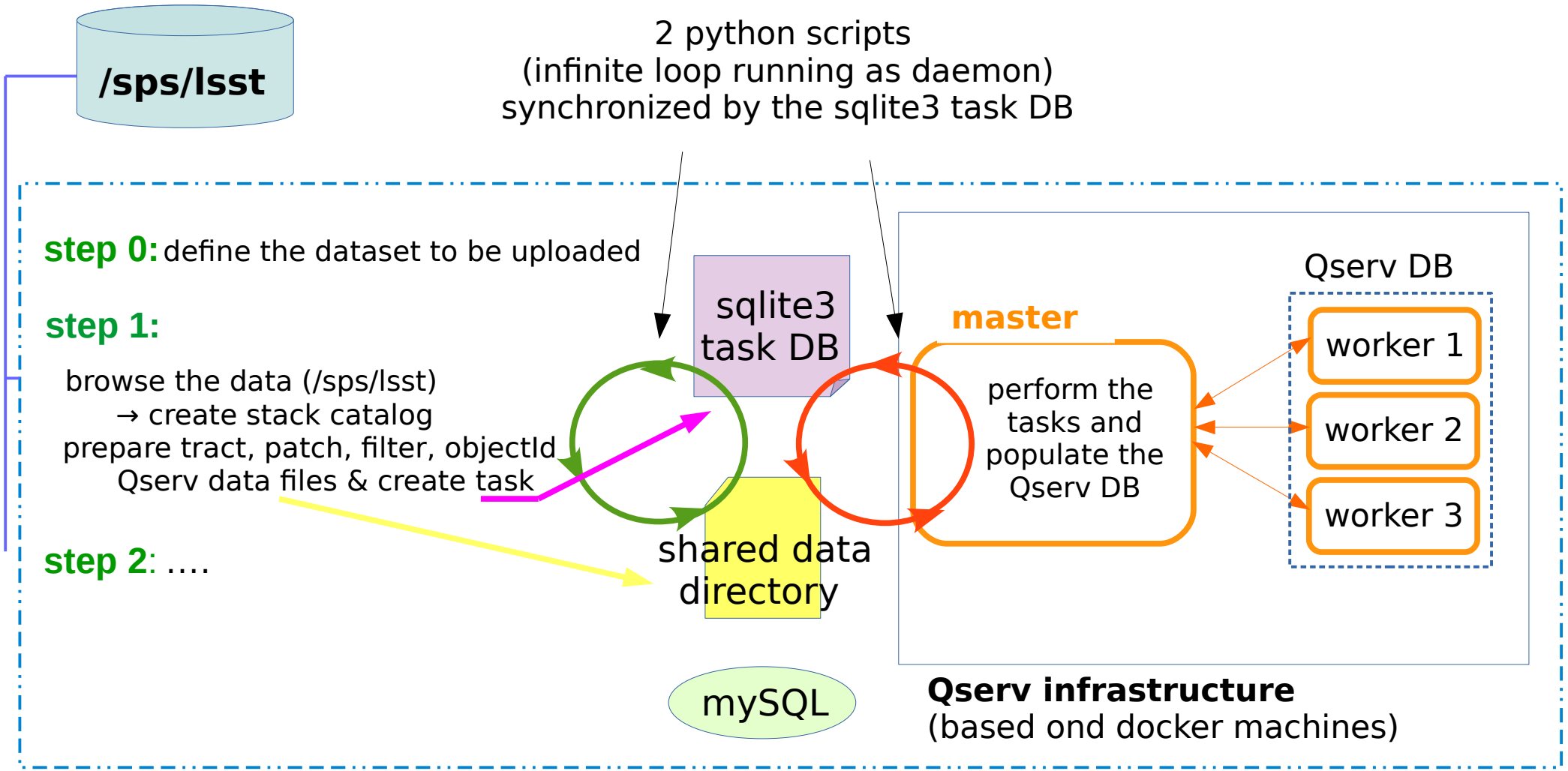
 avoid developing new scripts (issue with the maintenance over time)



- deployment worked out of the box (thanks to F. Jammes & N. Chotard)
- Qserv python upload scripts had to be modified to allow to ingest the data in a 2-step process (see previous slide)

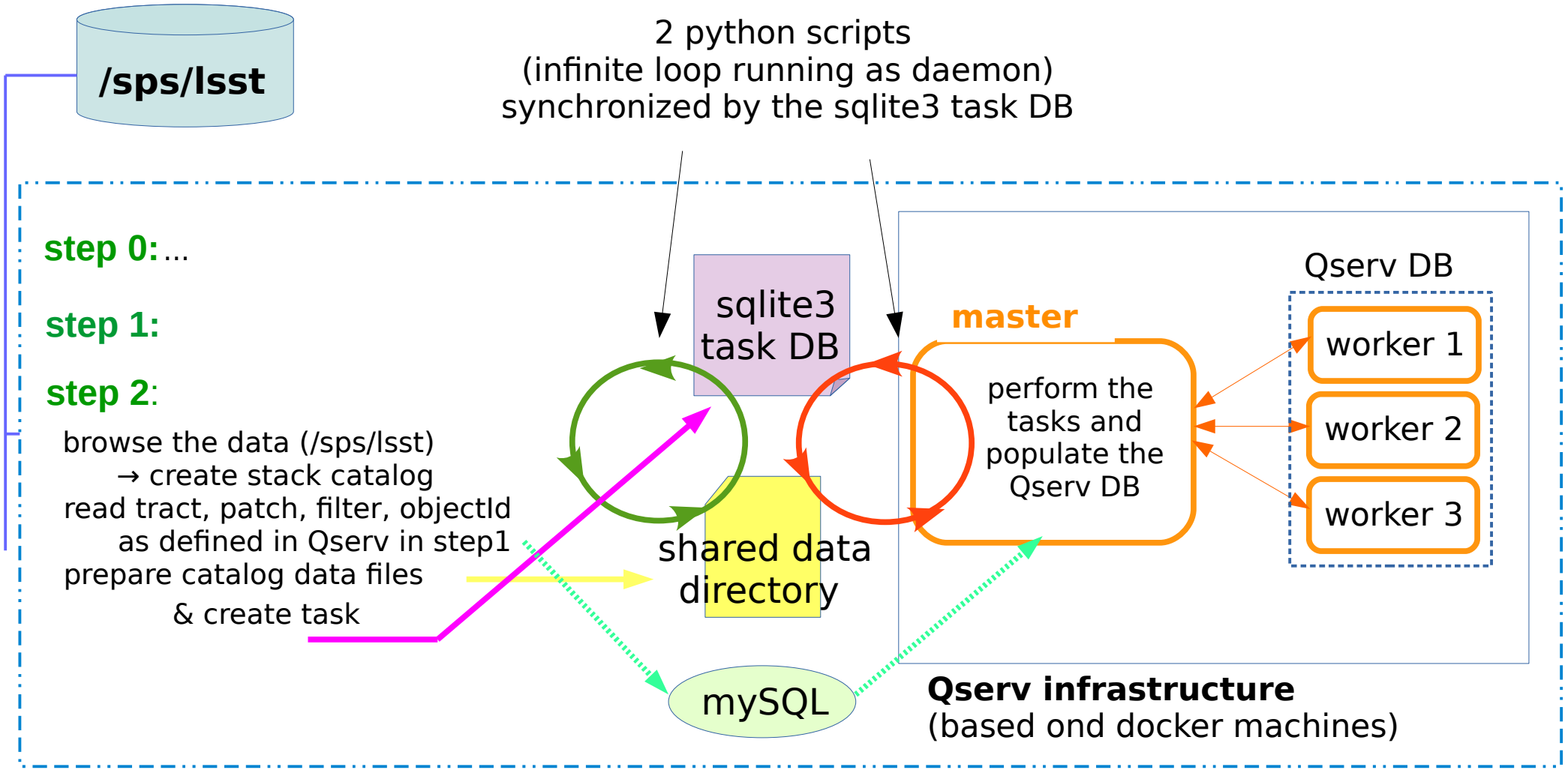
Current status and perspective

a first prototype is available



Current status and perspective

a first prototype is available - cont'd



Current status and perspective

Current status - cont'd

Parallelisation :

step1 : done by a single process

step2 : catalogs can be easily split following different sets of (filter, tract, patch) values

⇨ Qserv loading process run in parallel mode

Ongoing :

access the Qserv database from a CC interactive machine (ccage) using mysql requests (without remote ssh connexion)

Next steps

Ingest data reprocessed by N. Chotard

⇨ validation by comparison of analysis results obtained with original stack data vs Qserv DB data (N. Chotard)

No performance test made yet (load test, timing, ...)

Error recovery tests (use the sqlite3 task DB as input)

⇨ use Nicolas reprocessed data as input to complete these tasks