Astronomy ESFRI & Research Infrastructure Cluster
ASTERICS - 653477

Reprocessing of LSST precursor data sets at CC-IN2P3 with the LSST stack

Nicolas Chotard, Dominique Boutigny, Sabine Elles

LSST France @ CPPM

17th, January 2018

Data reprocessing

- Several catalogs created with the DM-stack are already available
  - SXDS data from HSC (/sps/lsst/dev/lsstprod/hsc/SXDS/output)
  - CFHT data containing clusters (/sps/lsst/data/clusters)
  - CFHT D3 field (/sps/lsst/data/CFHT/D3)
  - Others?
- Automated data reprocessing using the DM-stack also initiated
  - From several telescopes and instruments : CFHT, Decam, HSC, etc.
  - For different analyses : SNe, Clusters, etc.
  - Trying to be flexible enough to run on different data sets
- Two applications envisioned
  - Access to larger/better catalogs → essential step to the construction and validation of the analysis pipelines
    → Using stable versions → reprocessing of more data + analysis
  - Use and tests of the LSST DM-stack
    → Weeklies → scientific validation of the code (direct feedback to DM) + improvement of the catalog qualities

Status

- Complete reprocessing is now working
  - Use of the DESC workflow engine (SRS Pipeline)
  - One galaxy-cluster field (CFHT data) fully processed using a recent version of the DM-stack (`w_2017_49`)
  - Took about a day to process this data-set, was taking about a week before (when done manually)
- Mostly working on weekly releases of the DM stack
  - Encounter regular problems in making the stack working from end to end with the weeklies → Regular feedback given to DM-stack team
  - Using a small and controlled data-set make it easier and faster to give them feed-back for weekly releases
  - Works as one continuous integration/system tests

Status

- Currently reprocessing ~800 CFHT visits (~40 galaxy clusters)
  - Sparse data on the sky
  - Full rings sky-map created (as for HSC reprocessing) → does not depends on the input data-sets
  - Consistent reprocessing for these visits
  - Should be able to compute the mass of some of these clusters
- Getting ready to process HSC data to get more clusters
  - The pipeline is configurable
  - Should be ready to run on HSC (or other) data

Configuration

Automatic reprocessing is now configured with a simple file, and then launched from the SRS web server

```
# Default location from which to setup DM
export DM_RELEASE="w_2017_49"
export DM_SETUP="/sps/lsst/software/lsst_distrib/${DM_RELEASE}/loadLSST.bash"
# Configuration files for the DM scripts in
# Taken in /sps/lsst/users/lsstprod/ReprocessingTaskForce/config/${DM_CONFIG}
export DM_CONFIG="${DM_RELEASE}/cfht"
# Base directory for input and output data
export VISIT_DIR="/path/to/the/visits"                # inputs
export OUTPUT_DATA_DIR="/where/to/save/outputs”       # outputs
# The filters
# Must be compatible with the DM_CONFIG
export FILTERS="u,g,r,i,z"
```

Catalog access

Data reprocessing

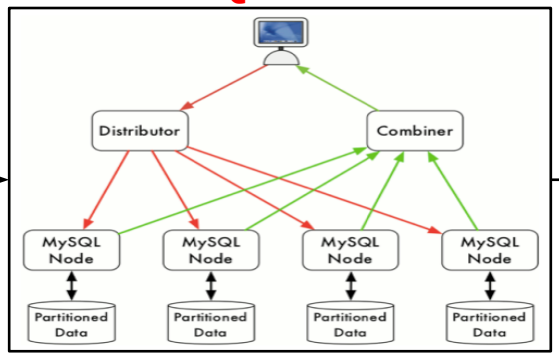
Data access

Data analysis

Automatized data reprocessing
(**LSST stack** + Workflow engine)

Catalogs

Qserv instance



Local files with data

Python interface

Local queries

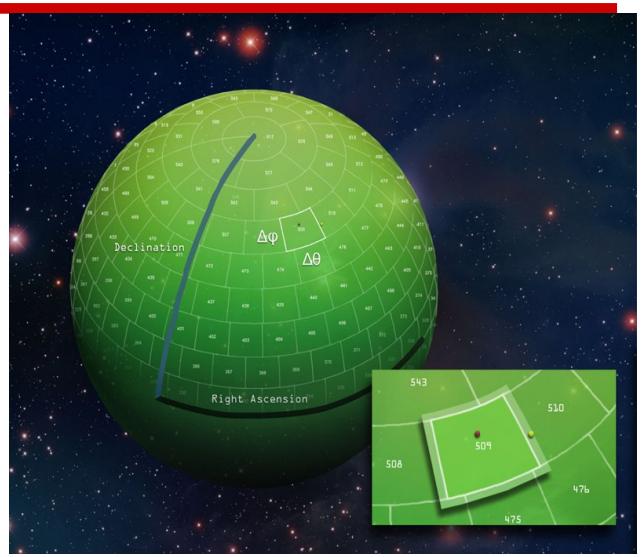
Distant queries

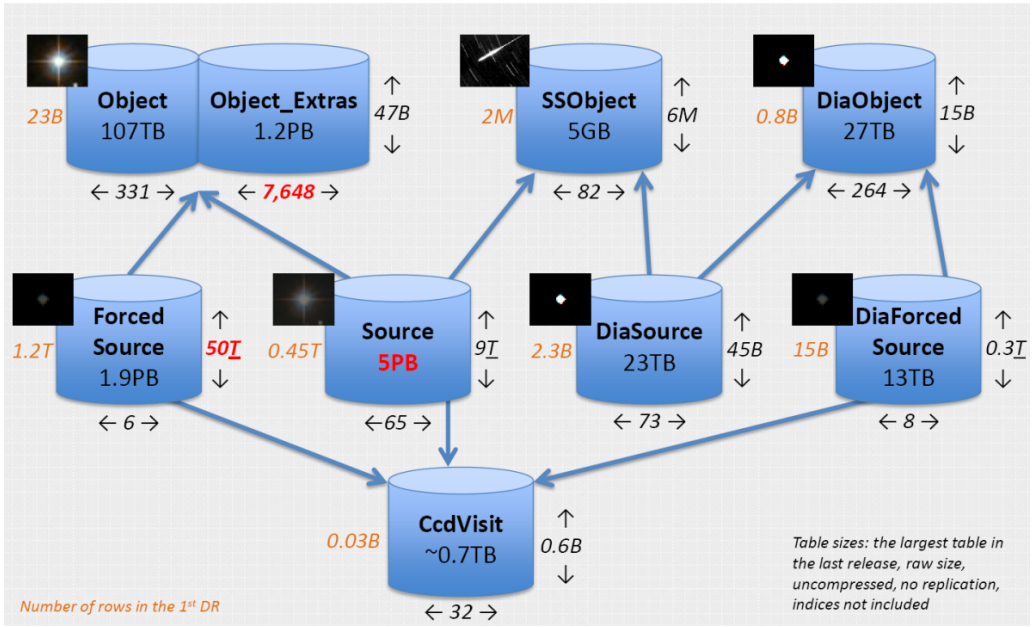
Science pipelines

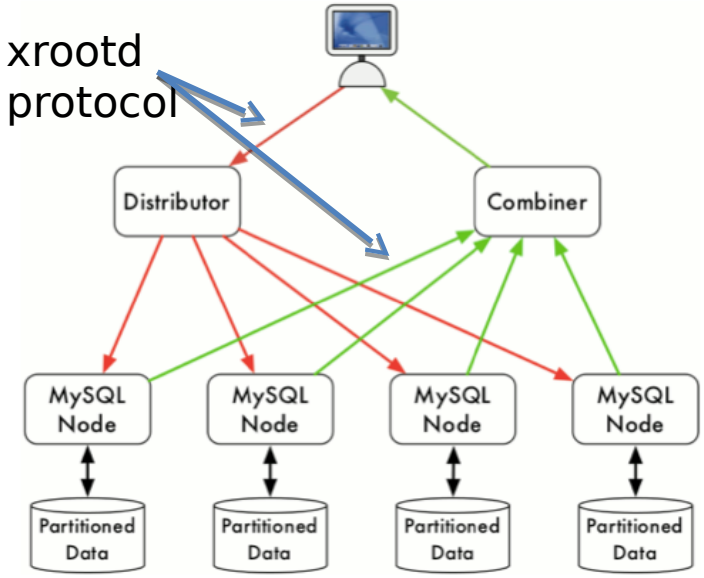
Qserv database

Challenge: design an open source SQL database system able to store trillions of objects while keeping a reasonable access time

Qserv: developed at SLAC + IPAC
Design optimized for astronomical queries



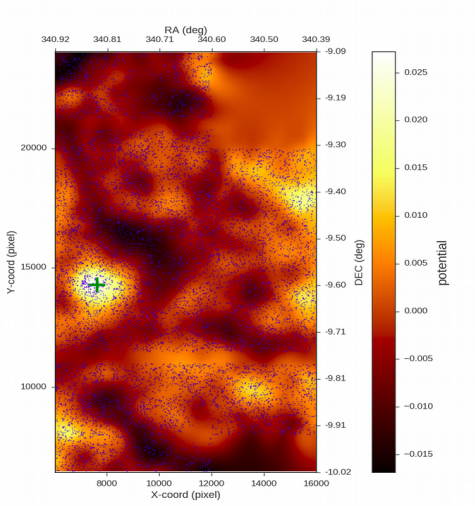




Massively parallel – distributed – fault tolerant **relational database**

Qserv integration

- Test Qserv on real data processed through the LSST software stack
  - Different queries (magnitudes, position, etc.)
  - Different configuration of the DB (number of stripes and chunks)
  - Different catalogs (sources, coadds)
  - ➢ Test its capabilities and performances
- Qserv integration into science analysis pipelines
  - Automatic inclusion of LSST stack-processed data into a Qserv instance
  - Direct queries in this database from a science pipeline
  - Construction/test of python tools to query the data
- ➔ Test case: Clusters pipeline
  - Galaxy cluster mass estimate
  - LSST stack data used in all steps of the analysis
  - CFHT data already processed
  - 5 filters, several areas of the sky



Status

- Short term
  - Set-up a test-case Qserv instance @CC-IN2P3 (OpenStack)
  - Incremental load of data (prototype already developed)
  - Reprocess and load more data (telescope, filters, sky area)
- Longer term
  - Permanent Qserv instance @CC-IN2P3 dedicated to analysis
  - Automatic and incremental ingestion of new LSST stack-processed data in the CC-IN2P3 Qserv instance
  - Python tools to query these data
- Implementation of these tools in DESC science pipelines
- Qserv tests and validation on real science cases in LSST/DESC

Status

- Short term
  - Set-up a test-case Qserv instance @CC-IN2P3 (OpenStack)
  - Incremental load of data (prototype already developed)
  - Reprocess and load more data (telescope, filters, sky area)
- Longer term
  - Perm… to analysis
  - Automatic and incremental ingestion of new LSST stack-processed data in the CC-IN2P3 Qserv instance
  - Python tools to query these data
- Implementation of these tools in DESC science pipelines
- Qserv tests and validation on real science cases in LSST/DESC

See next talk (Sabine Elles)

Data/software access

- In a distant future
  - Official LSST data access center tools
- In a near future
  - Jupyter hub at CC-IN2P3
  - Currently in development
- Today
  - stackyter
  - Local display of a Jupyter notebook running on a distant server

stackyter - goals

- Local display of a Jupyter notebook running on a distant server
  - Run the script in your terminal
  - The Jupyter server will be launch on the distant host
  - Open your local brower
  - Work on the distant host from your local brower
- Use cases
  - Can connect to any host as long as Python/Jupyter is available
  - Can set up your environment using a setup file on the host
  - Can set up the LSST stack and a DESC "environment" with access to catalogs

stackyter - usage

- Installation
  - “pip install stackyter”
- Usage
  - “stackyter.py [options]”
  - “Ctrl-C” to stop the Jupyter server and close the connection
- Main options
  - --host : to select the host to connect to (e.g., ccage.in2p3.fr)
  - --username : your username on this host
  - --mysetup : the path to your setup file (in which you define the path to Jupyter)
  - --workdir : the path to your working directory (it has to exist)

stackyter - @ CC-IN2P3

- 2 options have been made to work in a « LSST environment »
- stackyter.py --vstack v14.0
  - Setup of any version of the LSST stack
- stackyter.py --desc
  - “DESC environment”
  - A miniconda 3 installation
  - The GRC (Generic Catalog Reader) and grc-catalogs packages, to load and read the DESC catalogs
  - The **proto-dc2_v2.0** DESC catalogs

stackyter - configuration

- Configuring stackyter is optional
  - The configuration file contains « default » values for the script command line options
  - It does not need to be define to use the script
- Default configuration file can be located under
  - $HOME/.stackyter-config.yaml
  - $STACKYTERCONFIG
- Content
  - Several configuration can be defined (all named differently)
  - If more than one configuration is defined, a default one must be defined as well
  - --config : select the configuration you want to use (from the default file)
  - --showconfig : show all available config found in a default config file

stackyter - configuration

```
{
 'default_config': 'host1',

 'host1': {
           'host': 'myhost.domain.fr',
           'jupyter': 'lab',  # if installed
           'username': 'myusername',
           'mysetup': '/path/to/my/setup/file.sh',
           'workdir': '/path/to/my/directory/'
            },

 'host2': {
           'host': 'otherhost.fr',
           'username': 'otherusername',
           'mysetup': '/path/to/my/setup'
          },

 'stack': {
           'host': 'cca7.in2p3.fr',
           'packages': ["lsst_distrib"],
           'username': 'myusername',
           'vstack': 'v14.0',
           'workdir': '/pbs/throng/lsst/users/username/',
            },
 'desc': {
           'host': 'cca7.in2p3.fr',
           'username': 'myusername',
           'desc': True,
           'workdir': '/pbs/throng/lsst/users/username/'
          }
}
```

Summary

- LSST stack
  - Working and allowing us to process existing data sets and to produce catalogs ready for analysis
  - An automatic way to use it is available through the SRS work-flow engine system – also configurable
- Data access through Qserv
  - Outputs of the LSST stack will soon be available through query to a Qserv instance at CC-IN2P3
  - Starting to modify the Clusters pipeline to use this DB (instead if local files)
- Jupyter notebook for analysis
  - stackyter is a convenient way to use Jupyter notebook running on distant server
  - It is configurable and can be used for all host as long as Python/Jupyter is available
  - LSST/DESC “environment” has also been created
    - Output of the stack can easily be parsed and analyzed
    - Proto-dc2 catalog also available with tools to study it

What next?

- Test an other work-flow management system
  - Pegasus? DM SuperTasks ? An other tool?
- Include data validation steps at each stage of the data re-processing procedure
- Process more data from more sources
  - HSC data (wide field)
  - Combine CFHT and HSC data for cluster analysis
  - Other data sets/telescopes?
- Automatic ingestion of re-processed data into Qserv
- Adapt the Cluster pipeline to query Qserv