Cosmic Peta-Scale Data Analysis at IN2P3

Fabrice Jammes
Scalable Data Systems Expert
LSST Database and Data Access Software Developer

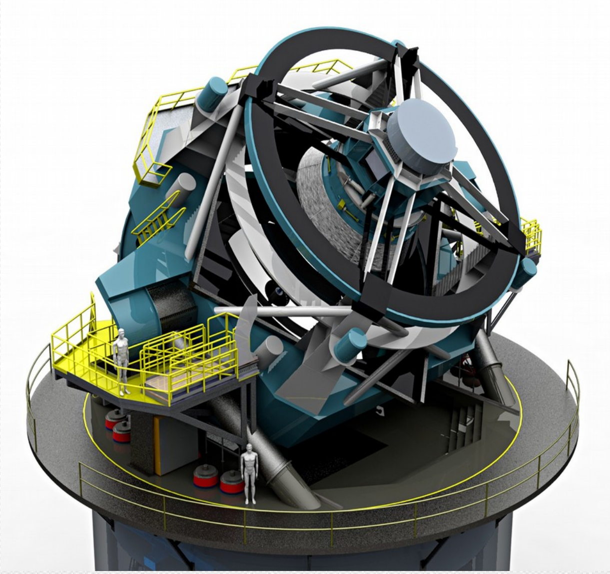
Yvan Calas
Senior research engineer
LSST deputy project leader at CC-IN2P3

Fabio Hernandez
Senior research engineer
LSST project leader at CC-IN2P3

Jacek Becla
SLAC Technology Officer for Scientific Databases
LSST Database and Data Access Manager

LSST in short

- ➢8.4 m telescope
- ➢Cerro Pachon (Chile)
- ➢(Very) wide-field astronomy
- ➢All visible sky in 6 bands ~20000□
- ➢15 s exposure, 1 visit / 3 days
- ➢During 10 years !
- ➢60 Pbytes of raw data



Who We Are

Andrew
Hanushevsky
0.4

Andrei
Salnikov
0.5

Brian Van
Klaveren
0.4

Jacek
Becla

John
Gates
1

Fabrice
Jammes
0.3 (+0.5)

**Fritz
Mueller**
1

Nate
Pease
1

Vaikunth
Thukral
(1)

Igor
Gaponenko
(1)

???
1

Who We Are: French Operation Team

CCIN2P3

Centre de Calcul
de l'Institut National de Physique Nucléaire
et de Physique des Particules

Yvan
Calas

Fabio
Hernandez

And others experts:
Fabien Wernli (Monitoring)
Loïc Tortay (GPFS),
Mathieu Puel (System administration)

What We Do

- Data Access and Database
- Data and metadata
- Images and databases
- Persisting and querying
- For pipelines and users
- Real time Alert Prod and annual Data Release Prod
- For Archive Center and all Data Access Centers
- For USA, France and international partners
- Persisted and virtual data
- Estimating, designing, prototyping, building, and productizing

Database Schema

LSST Database Schema Browser alpha

Large Synoptic Survey Telescope

Schema versions available for browsing: **baseline** | DC3a | PT1_1 | PT1_2 | ImSim | S12_sdss | S12_lsstsim (underlined showed)

User defined functions documentation: version 0.1, version 0.2, version 0.3 (default on lsst10)

Table List

- AAA_Version_3_2_4
- ApertureBins
- CcdVisit
- CcdVisitMetadata
- DiaForcedSource
- DiaObject
- DiaObject_To_Object_Match
- DiaSource
- ForcedSource
- LeapSeconds
- Object
- Object_APMean
- Object_Extra
- Object_NonPeriodic
- Object_Periodic
- prv_Amp
- prv_Ccd
- prv_cnf_Amp
- prv_cnf_Ccd
- prv_cnf_Filter
- prv_cnf_Fpa
- prv_cnf_InputDataSet
- prv_cnf_Node
- prv_cnf_Raft
- prv_cnf_Run
- prv_cnf_Task
- prv_cnf_Task2TaskExecution
- prv_cnf_Task2TaskGraph
- prv_cnf_TaskExecution
- prv_cnf_TaskGraph
- prv_cnf_TaskGraph2Run
- prv_Filter
- prv_Fpa
- prv_InputDataSet
- prv_Node
- prv_ProcHistory
- prv_Raft
- prv_Run
- prv_Snapshot
- prv_Task

Details for table *Object*

The Object table contains descriptions of the multi-epoch static astronomical objects, in particular their astrophysical properties as derived from analysis of the Sources that are associated with them. Note that fast moving objects are kept in the MovingObject tables. Note that less-frequently used columns are stored in a separate table called Object_Extra.

name	type	not null	unit	ucd	description
objectId	BIGINT	y		meta.id;src	Unique id.
parentObjectId	BIGINT				Id of the parent object this object has been deblended from, if any.
procHistoryId	BIGINT	y			Pointer to ProcessingHistory table.
psRa	DOUBLE		deg	pos.eq.ra	RA-coordinate of the center of the object for the Point Source model at time 'psEpoch'.
psRaSigma	FLOAT		deg	stat.error;pos.eq.ra	Uncertainty of psRa.
psDecl	DOUBLE		deg	pos.eq.dec	Decl-coordinate of the center of the object for the Point Source model at time 'psEpoch'.
psDeclSigma	FLOAT		deg	stat.error;pos.eq.dec	Uncertainty of psDecl.
psMuRa	FLOAT		mas/yr	pos.pm	Proper motion (ra) for the Point Source model.
psMuRaSigma	FLOAT		mas/yr	stat.error;pos.pm	Uncertainty of psMuRa.
psMuDecl	FLOAT		mas/yr	pos.pm	Proper motion (decl) for the Point Source model.
psMuDeclSigma	FLOAT		mas/yr	stat.error;pos.pm	Uncertainty of psMuDecl.
psParallax	FLOAT		mas	pos.parallax	Stellar parallax. for the Point Source model.
psParallaxSigma	FLOAT		mas	stat.error;pos.parallax	Uncertainty of psParallax.
uPsFlux	FLOAT		nmgy	phot.count	Calibrated flux for Point Source model for u filter.
uPsFluxSigma	FLOAT		nmgy	stat.error;phot.count	Uncertainty of uPsFlux.
gPsFlux	FLOAT		nmgy	phot.count	Calibrated flux for Point Source model for g filter.
gPsFluxSigma	FLOAT		nmgy	stat.error;phot.count	Uncertainty of gPsFlux.
rPsFlux	FLOAT		nmgy	phot.count	Calibrated flux for Point Source model for r filter.

http://ls.st/s91

Data

Images

Persisted: **~38 PB**
Temporary: **~½ EB**

~3 million “visits”
~47 billion“objects”
~9 trillion “detections”

Largest table: **~5 PB**
Tallest table: ~50 trillion rows
Total (all data releases, compressed): **~83 PB**

Ad-hoc user-generated data
Rich provenance

Production Data

- **Database**
  - **Real-time Alert DB**.
    No-overwrite updates between Data Releases
    Real-time replica of Alert Prod DB for analytics. No long-running analytics here
  - **Immutable Database (+user workspaces)**
    Released annually. Immutable
    2 most recent releases on disk

- **Images**
  - raw: 2 most recent visits for each filter
  - coadds and templates: for 2 most recent releases
  - raw calibration: most recent 30 days
  - science calibrated: most recent 30 days
  - observatory telemetry: all
  - cutouts for alerts: all
  - EPO full-sky jpeg: one set

Analytics

- **Aiming to enable majority of analytics via database**
- **Aiming to enable rapid turnaround on exploratory queries**

- **In a region**
  - get an object or data for small area - <10 sec
- **Across entire sky**
  - Scan through billions of objects - ~1 hour
  - Deeper analysis (Object_*) - ~8 hours
- **Analysis of objects close to other objects**
  - ~1 hour, even if full-sky
- **Analysis that requires special grouping**
  - ~1 hour, even if full sky
- **Time series analysis**
  - Source, ForcedSource scans - ~12 hours
- **Cross match & anti-cross match with external catalogs**
  - ~1 hour

Sizing the system for ~100 interactive + ~50 complex simultaneous DB queries.
Same for images

APIs

- Metadata
  - RESTful WebServ
- Images
  - RESTful ImageServ
- Databases
  - RESTful DbServ
  - SQL92 +/-, MySQL-like DBMS
  - Next-to-database python-based
- Query volume controlled by Resource Mgmt

Additions (“SQL92 +”)

➢ **Spatial constraints**
- qserv_areaspec_box(lonMin, latMin, lonMax, latMax)
- qserv_areaspec_circle(lon, lat, radius)
- qserv_areaspec_ellipse(semiMajorAxisAngle, semiMinorAxisAngle, posAngle)
- qserv_areaspec_poly(v1Lon, v1Lat, v2Lon, v2Lat, ...)

```
SELECT objectId
FROM   Object
WHERE  qserv_areaspec_box(2,89,3,90)
```

Current Restrictions (SQL92 +)

Only a SQL subset is supported

For example:

- Spatial constraints (must use User Defined Functions, must appear at the beginning of WHERE, only one spatial constraint per query, arguments must be simple literals, OR not allowed after area qserv_areaspec_*)
- Expressions/functions in ORDER BY clauses are not allowed
- Sub-queries are NOT supported
- Commands that modify tables are disallowed
- MySQL-specific syntax and variables not supported
- Repeated column names through * not supported

Selected Common Query Types

- SELECT sth FROM Object
  – massively parallel

- SELECT sth FROM Object WHERE qserv_areaspec_box(....)
  – selection inside chunks that cover requested area, in parallel

- SELECT sth FROM Object JOIN SOURCE USING (objectId)
  – massively parallel without any cross-node communication

- SELECT sth FROM Object WHERE objectId = <id>
  – quick selection inside one chunk

Common queries – see http://ls.st/ed4

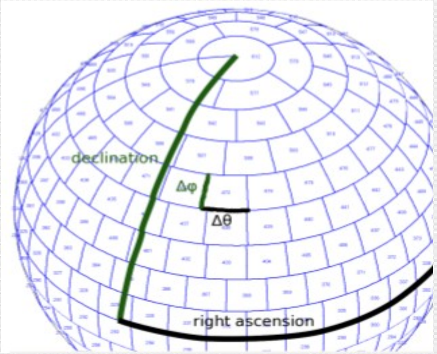
QServ Under the Hood

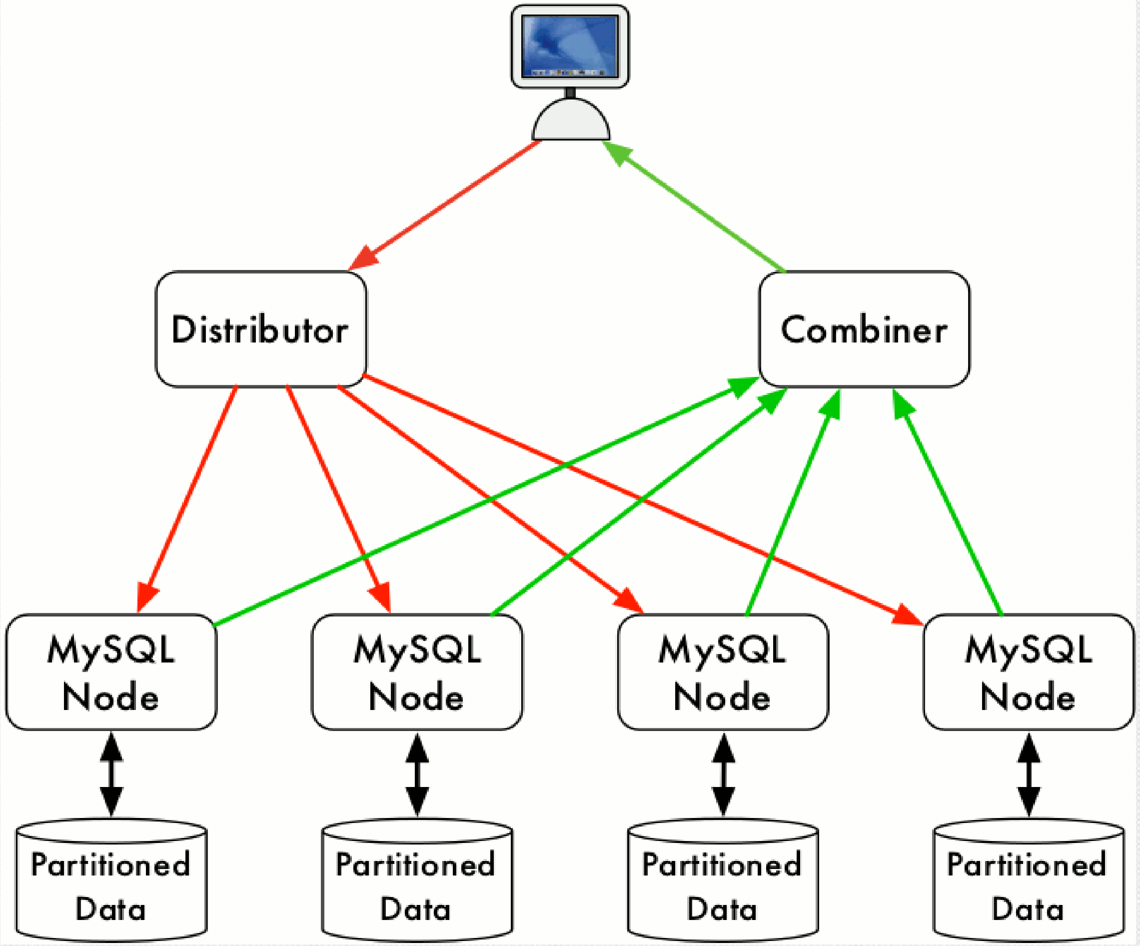
Implementation Strategy

- 100% Open source
- Keep it flexible
- Hide complexity
- Reuse existing components:
  MariaDB, MySQL Proxy, XRootD, Google protobuf, flask
- Plus custom glue
  - C++ + a bit of python. Some ANTLR
  - Lots of multithreading, callbacks, mutexes and sockets
- And custom UDFs

QServ Design

- ➢Relational database, spatially-sharded with overlaps
- ➢Map/reduce-like processing





Key Features

- **Scalable spherical geometry**
  - 0/360 RA wrap around, pole distortion, convex polygons,
  - accurate distance computation, functions for distance (angle),
  - point-in-spherical-region tests (circle, ellipse, box, convex polygon)
  - Custom (HTM-based) UDFs (https://github.com/wangd/scisql)
- **Optimized spatial joins for neighbor queries, cross-match**
  - Spherical partitioning with overlap
  - Director table, secondary index
  - Two-level, 2nd level materialized on-the-fly
- **Shared scans**
  - Continuous, sequential scans through data, including L3 distributed tables
  - (Non-interactive) queries attached to appropriate running scan
- **All internal complexity transparent to end-users**

Tests and Demonstrations

300 nodes, 10 TB data set

- 1-4 sec easy queries, 10 sec-10 min table scans, ~5 min complex joins

Running now: 2x 25 nodes, ~35 TB data set @IN2P3

+ LVM-express machine with ~2TB memory

In the near term: **Prototype Data Access Center at NCSA**

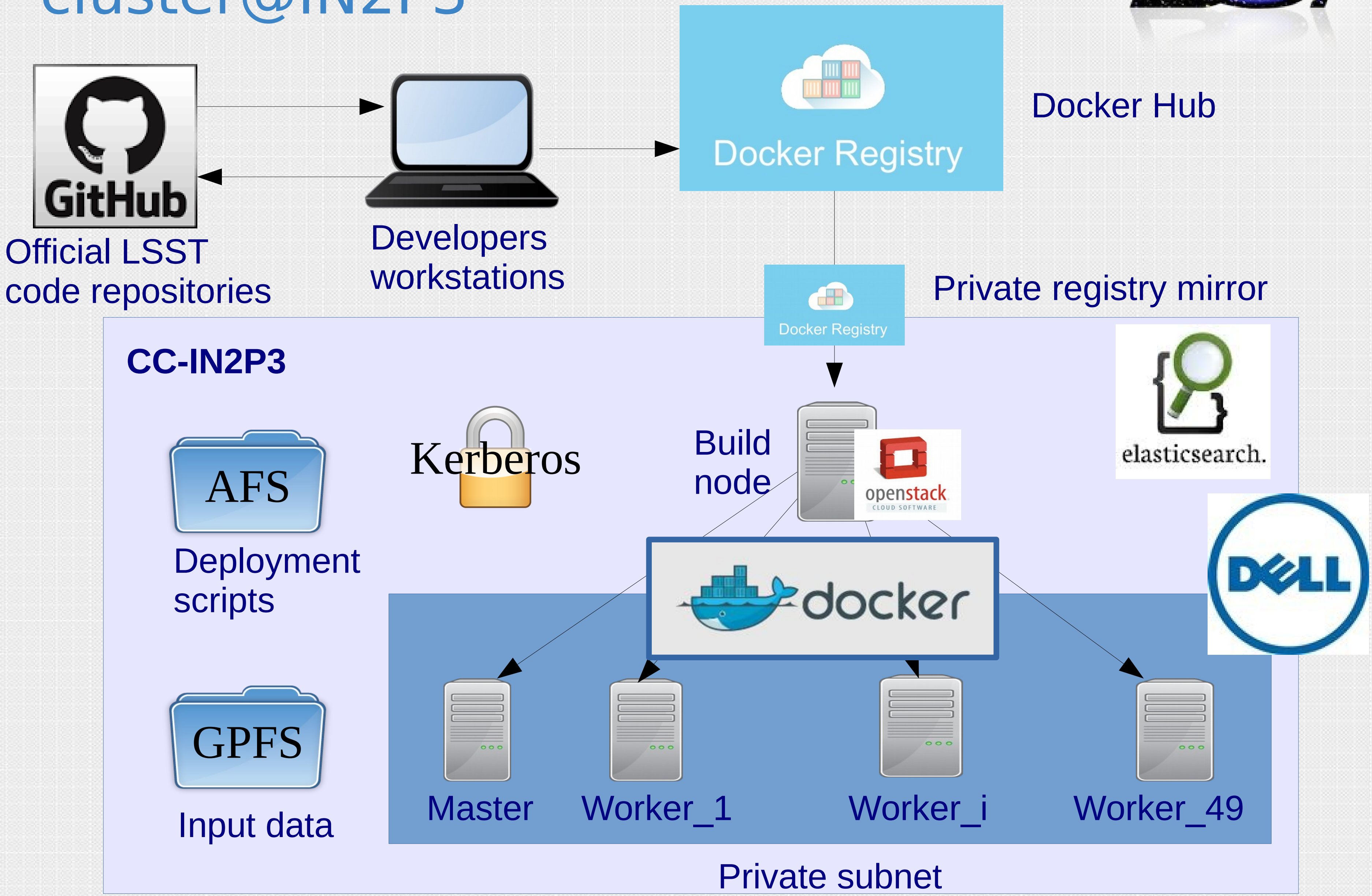
Scale testing to date @IN2P3

S15 large scale tests:
Data: replicated SDSS Stripe 82
~10% DR1 (~2B Object, ~35B Source, ~172B F. Source)
Hardware: 24 nodes @ IN2P3, 2 x 1.8GHz 4 core, 16G RAM

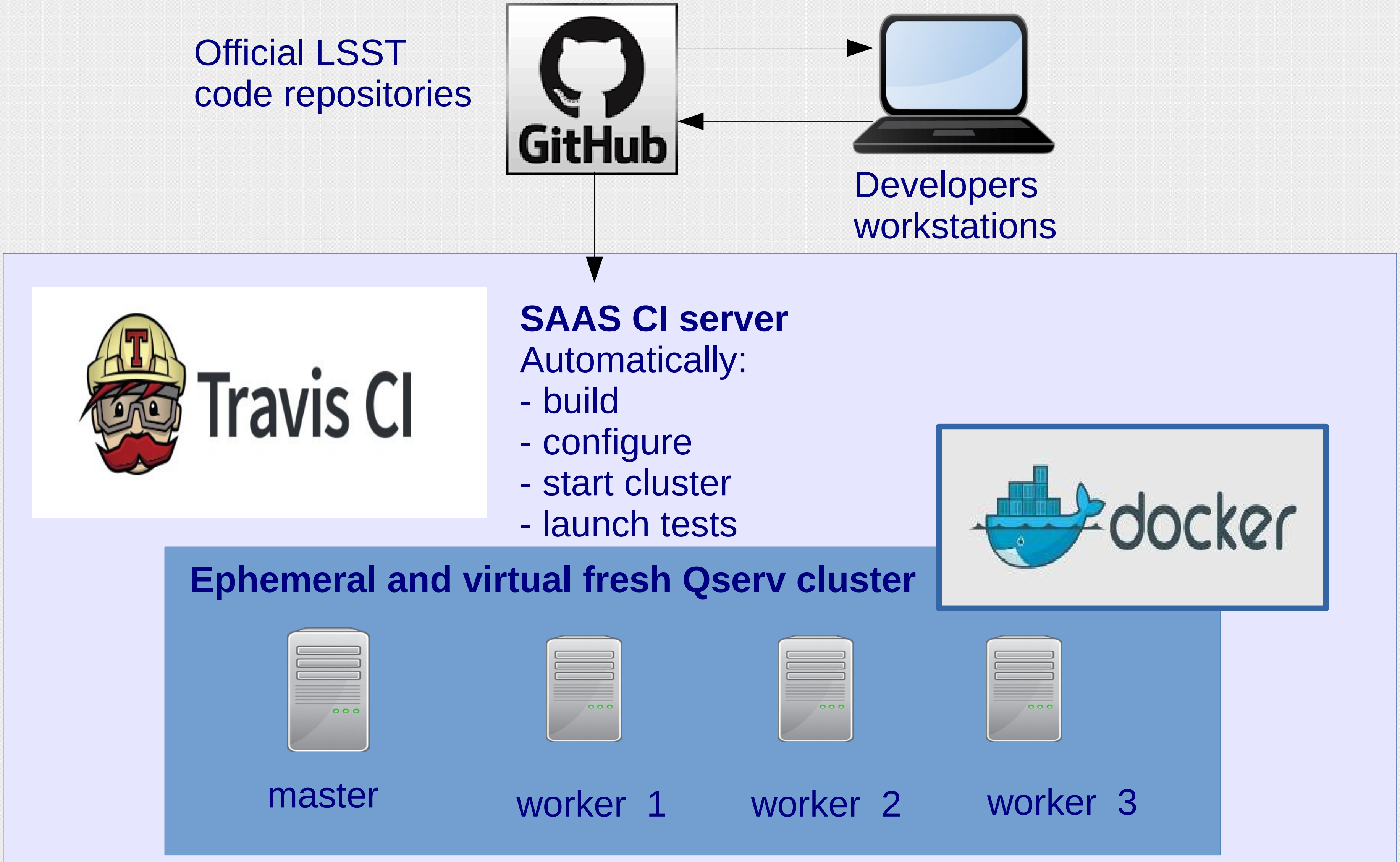
Simul. 50 low-volume queries + 5 high-volume queries:
<1s for low-volume queries
~15m for high-volume Object scans
~1h for Source scans

See confluence page “S15 Large Scale Tests”

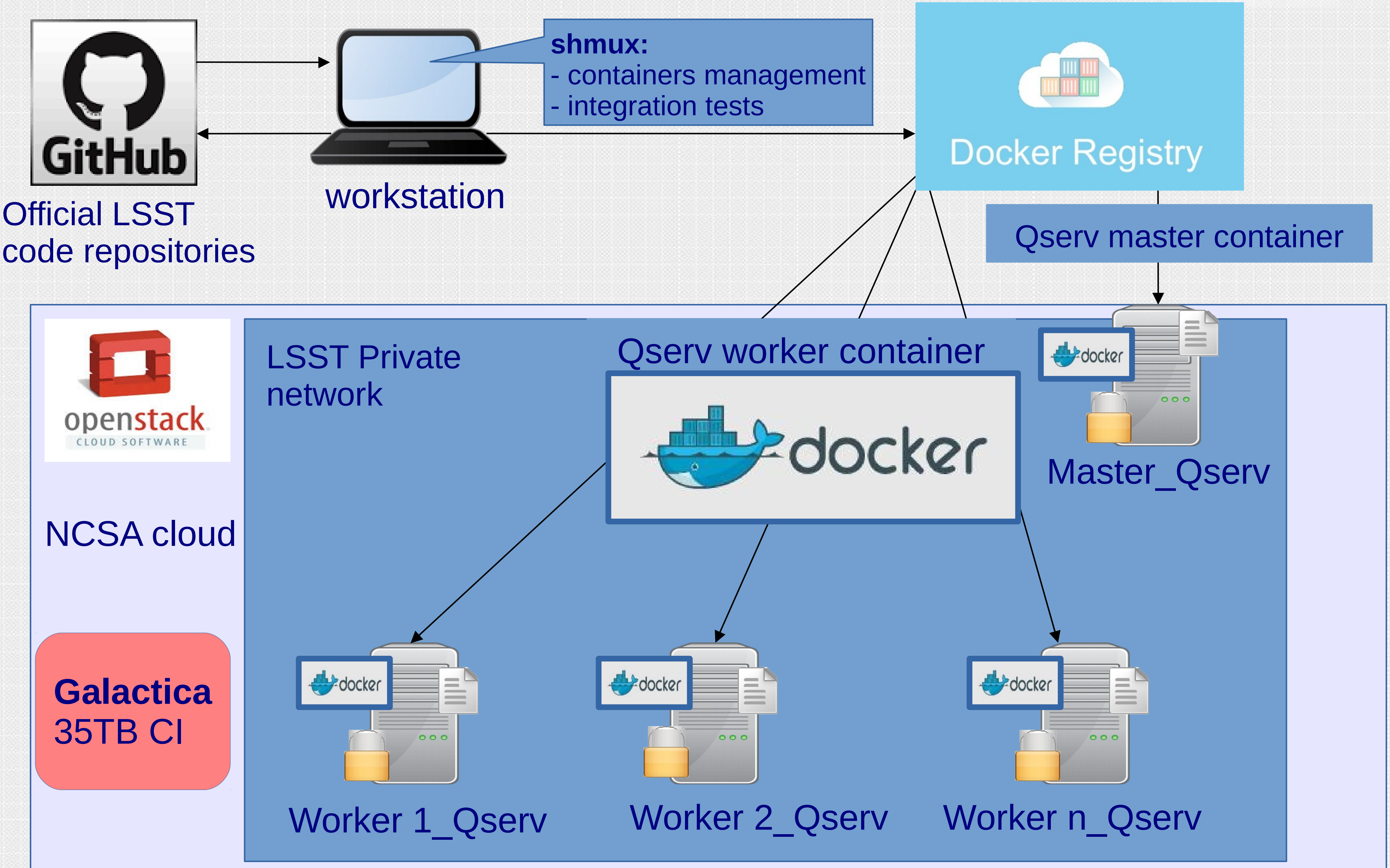
cluster@IN2P3



CI multi-node integration tests



Automated Qserv deployment in OpenStack



Summary

- Big Data with Complex Analytics
- Spatially-sharded, map/reduce-like RDBMS
- Open source + custom glue
- Optimized for astronomical data sets at scale
- Have working prototype
- Turning it into a production system
- Want to learn more?
  - http://ls.st/4gh (Database Design doc)
  - http://ls.st/6ym (User Manual)
- Are you an adventurous super early adopter? You can try it now
  - http://ls.st/89y (Qserv Documentation)

Implementation Details

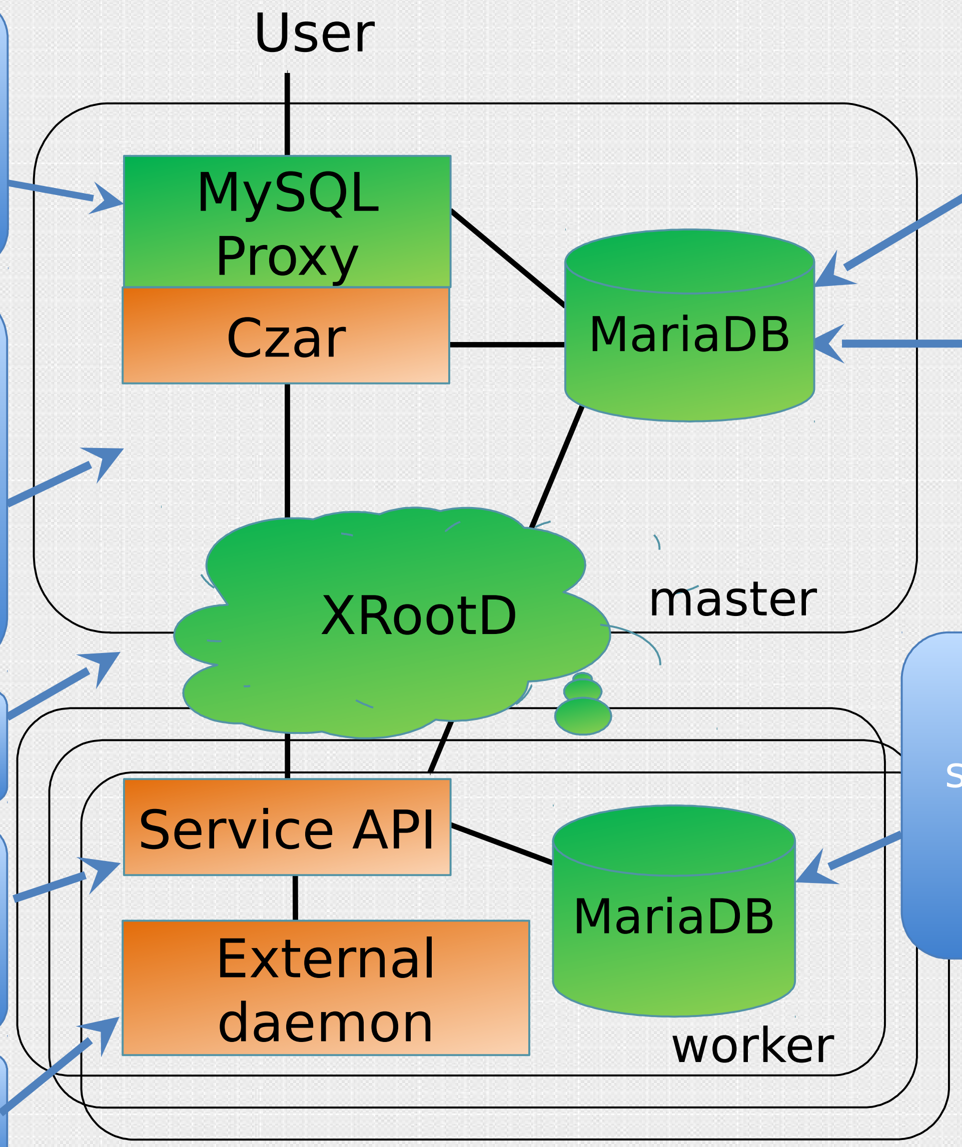
Intercepting user queries
Near-standard SQL subset with a few extensions

Query parsing and fragmentation generation, worker dispatch, spatial indexing, query recovery, optimizations, scheduling, result aggregation

Communication, replication

MariaDB dispatch, shared scanning, optimizations, scheduling

Specialized, non-SQL analytics



Result cache

Cluster control and configuration store

Single node RDBMS. Basic scanning, filtering, computation, aggregation, and joins