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 (+0.7)

Fritz Mueller
1

Mike Kelsey
0.5

Nate Pease
1

Serge Monkewitz
0.5

Vaikunth Thukral
(1)

???
1
Who We Are: French Operation Team

And others experts: 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

#### Table List

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>not null</th>
<th>unit</th>
<th>ucd</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectID</td>
<td>BIGINT</td>
<td>y</td>
<td>metadata</td>
<td></td>
<td>Unique ID.</td>
</tr>
<tr>
<td>parentObjectID</td>
<td>BIGINT</td>
<td>y</td>
<td></td>
<td></td>
<td>Id of the parent object this object has been debiased from, if any.</td>
</tr>
<tr>
<td>procHistoryID</td>
<td>BIGINT</td>
<td>y</td>
<td></td>
<td></td>
<td>Pointer to ProcessingHistory table.</td>
</tr>
<tr>
<td>psRa</td>
<td>DOUBLE</td>
<td>deg</td>
<td>pos.eq.ra</td>
<td></td>
<td>RA-coordinate of the center of the object for the Point Source model at time 'psEpoch'.</td>
</tr>
<tr>
<td>psRaSigma</td>
<td>FLOAT</td>
<td>deg</td>
<td></td>
<td></td>
<td>Uncertainty of psRa.</td>
</tr>
<tr>
<td>psDec</td>
<td>DOUBLE</td>
<td>deg</td>
<td>pos.eq.dec</td>
<td></td>
<td>Dec-coordinate of the center of the object for the Point Source model at time 'psEpoch'.</td>
</tr>
<tr>
<td>psDecSigma</td>
<td>FLOAT</td>
<td>deg</td>
<td></td>
<td></td>
<td>Uncertainty of psDec.</td>
</tr>
<tr>
<td>pmRa</td>
<td>FLOAT</td>
<td>mas/yr</td>
<td>pos.pm</td>
<td></td>
<td>Proper motion (ra) for the Point Source model.</td>
</tr>
<tr>
<td>pmRaSigma</td>
<td>FLOAT</td>
<td>mas/yr</td>
<td></td>
<td></td>
<td>Uncertainty of pmRa.</td>
</tr>
<tr>
<td>pmDec</td>
<td>FLOAT</td>
<td>mas/yr</td>
<td>pos.pm</td>
<td></td>
<td>Proper motion (dec) for the Point Source model.</td>
</tr>
<tr>
<td>pmDecSigma</td>
<td>FLOAT</td>
<td>mas/yr</td>
<td></td>
<td></td>
<td>Uncertainty of pmDec.</td>
</tr>
<tr>
<td>psParallax</td>
<td>FLOAT</td>
<td>mas</td>
<td>pos.parallax</td>
<td></td>
<td>Stochastic parallax for the Point Source model.</td>
</tr>
<tr>
<td>psParallaxSigma</td>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td>Uncertainty of psParallax.</td>
</tr>
<tr>
<td>pPSFlux</td>
<td>FLOAT</td>
<td>nmgY</td>
<td>phot.count</td>
<td></td>
<td>Calibrated Flux for Point Source model for u filter.</td>
</tr>
<tr>
<td>pPSFluxSigma</td>
<td>FLOAT</td>
<td>nmgY</td>
<td></td>
<td></td>
<td>Uncertainty of pPSFlux.</td>
</tr>
<tr>
<td>pPsFlux</td>
<td>FLOAT</td>
<td>nmgY</td>
<td>phot.count</td>
<td></td>
<td>Calibrated Flux for Point Source model for g filter.</td>
</tr>
<tr>
<td>pPsFluxSigma</td>
<td>FLOAT</td>
<td>nmgY</td>
<td></td>
<td></td>
<td>Uncertainty of pPsFlux.</td>
</tr>
<tr>
<td>pPsfFlux</td>
<td>FLOAT</td>
<td>nmgY</td>
<td>phot.count</td>
<td></td>
<td>Calibrated Flux for Point Source model for r filter.</td>
</tr>
</tbody>
</table>

---

**http://ls.st/s91**
~3 million “visits”
~47 billion “objects”
~9 trillion “detections”

Persisted: ~38 PB
Temporary: ~½ EB

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
  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
User Workspaces

➢ File system space
  – For images, configurations, software

➢ **Database user space**
  – For query results, external data
  – Co-located with immutable data
  – Distributed read-only, or non-distributed updatable

➢ Data size and access controlled by Resource Mgmt
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
Key Challenges (DB Team Perspective)

➢ Unknown unknowns & changing requirements
➢ Update for user-space data
➢ Provenance (traceability of temporary data)
➢ Data distribution @scale
➢ Spherical geometry
➢ Certain classes of queries: near neighbor, nonSql-ish analysis like time series
Design Philosophy

➢ 100% Open source
➢ Reuse what we can, build the rest
➢ Keep it flexible
➢ Hide complexity
➢ Build to scale beyond baseline
QServ Design

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

➢ Reusing 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
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

MySQL Proxy
Czar
MariaDB
XRootD master
Service API
External daemon
MariaDB worker

Result cache
Cluster control and configuration store
Single node RDBMS. Basic scanning, filtering, computation, aggregation, and joins
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
Current Status

➢ Working prototype
➢ Usable, but not bullet-proof
➢ Subset of features implemented, e.g., sub-queries not working
Recent work

➢ Query Executive improvements
➢ Shared Scan improvements
➢ Metadata system improvements
➢ Build, Packaging, and Test improvements
➢ Data Distribution prototyping
➢ CI multi-node integration tests
CI multi-node integration tests

Official LSST code repositories

GitHub

Developers workstations

SAAS CI server
Automatically:
- build
- configure
- start cluster
- launch tests

Ephemeral and virtual fresh Qserv cluster

master
worker 1
worker 2
worker 3
Upcoming work

➢ Documentation updates
➢ Shared scan work continues
➢ Data distribution work continues
➢ Large result-set improvements
➢ Secondary index improvements
➢ Pan-STARRS data
➢ Better query coverage

**In the long-term:** scalable data loading, data distribution, replica management, query management, resource management, user table support, non-SQL queries, and more. Plus productizing
Tests and Demonstrations

➢ 300 nodes, 10 TB data set
   – 1-4 sec easy queries, 10 sec-10 min table scans, ~5 min complex joins
➢ 20 nodes, 100 TB data set
➢ Concurrency
   – up to 100K in-flight chunk-queries, on ~100 nodes
➢ Fault tolerance
   – catching errors, transparent fail over to a replica
➢ Shared scanning
   – 30-query scan: 5m27s, avg speed for a single query: 3m
➢ RDBMS-agnosticity
   – Limited tests with MonetDB in place of MySQL
➢ Running now: 2x 25 nodes, ~35 TB data set @IN2P3
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”
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](http://ls.st/4gh) (Database Design doc)
   – [http://ls.st/6ym](http://ls.st/6ym) (User Manual)
➢ Are you an adventurous super early adopter? You can try it now
   – [http://ls.st/89y](http://ls.st/89y) (Qserv Documentation)