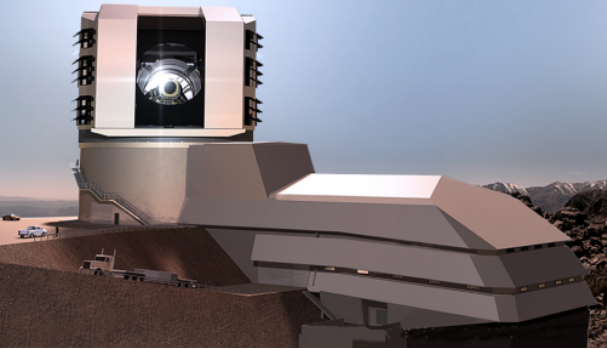
High-throughput computation in the LSP

Colin Slater, UW/LSST
Deputy DM Subsystem Scientist

LSP Design Review
April 10th 2019 Tucson, AZ



Scope of the Challenge

- One of the "classical" challenges of LSST is that data $\gg$ bandwidth.
- `Object` table is 50TB after one year and 300 TB at full-width; `ForcedSource` grows 65 TB per year.
- For comparison, 1 Gbps is $\sim$ 0.5 TB/hour.

- The logical solution is to adopt a "Bring your code to the data" model; circumventing the bandwidth bottleneck.
- DM was designed from the start to include user-driven processing inside the LSST datacenter.

Historical Digression

- Historical digression: User-driven processing was originally called "Level 3 Processing", and encompassed both image and catalog processing. "Next-to-the-database processing" was an implementation concept for user-driven, beyond-SQL, processing of catalogs.
- The boundary conditions of the problems these systems were designed to solve have shifted substantially since the LSST Final Design Review, so I will describe solutions that address these needs but don't quite fit into the boxes that were originally envisioned.

Formal Requirements

- LSR-REQ-0041, -0106, -0107 require that we support "Level 3 Data Products", which must interoperate with other LSST-produced products.
- LSR-REQ-0055 is the origin of the "10%" fraction for external user computing.
- DMS-REQ-0125 requires the project to provide a "software framework that facilitates" user-driven catalog processing, ensuring consistency and completeness. DMS-REQ-0128 is similar for images.
- The requirements are not very prescriptive about what the framework should look like.

Use Cases

- `http://dmtn-086.lsst.io` contains example use cases for high-throughput computations. E.g.:
  - Making stellar density maps — GROUP BY-like aggregations (Qserv)
  - Light curve fitting for interesting objects — (Jupyter)
  - Light curve feature computation for $\gtrsim 1\%$ of objects — challenging.
  - 2-point correlation functions — challenging.
  - Training machine learning models — challenging.
- These span a spectrum of users and use-cases, and for which we have a spectrum of capabilities.

Bring your code to the data

- GROUP BY-like aggregations: anything that can be done in SQL, should be in SQL.
  - Qserv gives us the ability to serve a large load of users with minimum resources.
  - The power of SQL is that each user's queries are entirely open to the infrastructure, so we can peek inside, add appropriate optimizations, perform sophisticated scheduling, and ensure correctness of results.

Bring your code to the data

- Fitting interesting objects: ideal for Jupyter Notebooks.
  - Necessary when analysis can't be formulated in SQL.
  - JupyterHub has made "Bring your code to the data" totally mainstream.
  - Better than was envisioned at the start of construction, because of the breadth of capabilities and the ease of use.
  - Seamlessly supports both first-time users and advanced analysis.

Bring your code to the data

- *Intensive* operations require something beyond this, scaling up both throughput and CPU.
- Correlation functions, fitting a substantial fraction of all light curves, ML training.
- Many industry-standard platforms for enabling user code to run on large datasets in a distributed environment. Hadoop, Dask, Spark, to name a few.
- Columnar Parquet files stored on an object server or shared filesystem is becoming a *de facto* standard for "big data" tools.

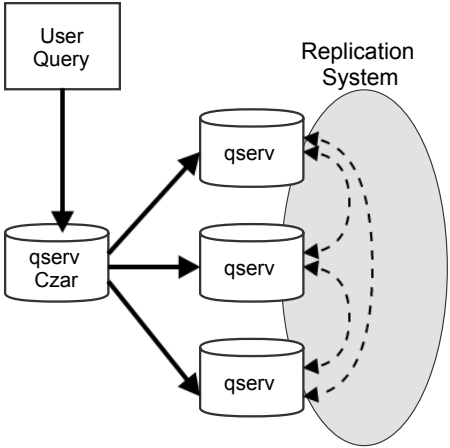
Bring your code to the data

- Dask/Spark are designed to let users maximize their available resources (bandwidth and CPU); the argument is that lots of commodity hardware can compete with more expensive specialized DB solutions.
- These frameworks enable exactly the computations what we want, but their resource strategy is exactly what we don't want
- The challenge for the project is determining how to support the computation users want without destroying the budget.

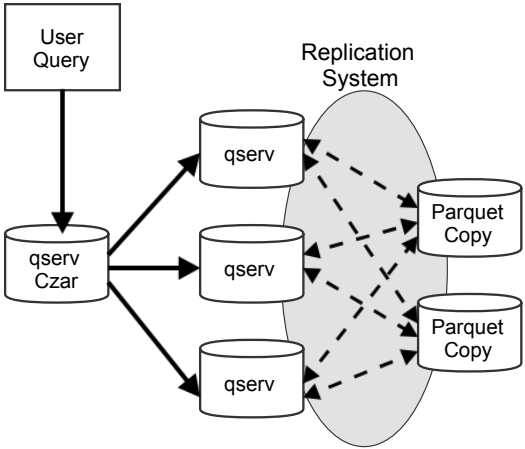
Technical progress

- We have made progress on the technical side of this.
- We have tested conversion of data tables between Parquet and Qserv's Mariadb tables; we need extra copies for recovery from node failures, so we can use Parquet for that and for access by "big-data" frameworks.

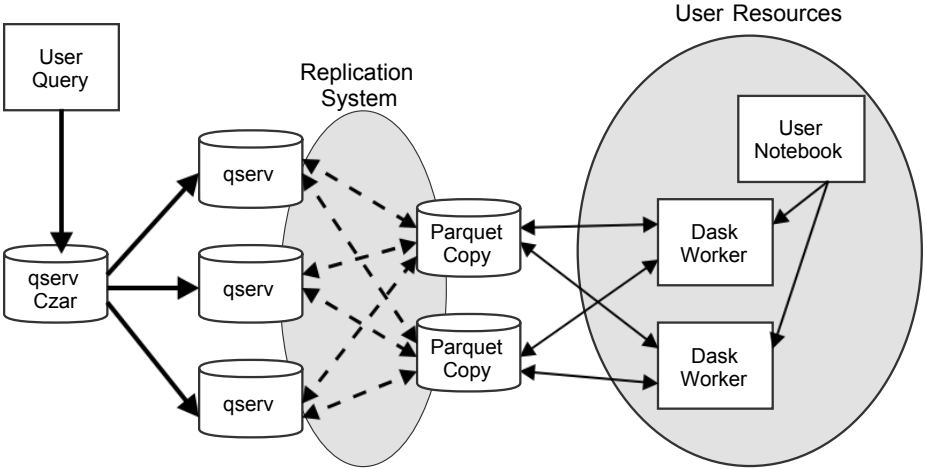
Architecture



Architecture



Architecture



Technical progress

- We have tested Dask for commissioning-era visualization and analysis needs; smaller datasets but we understand how to interface that with the notebook environment.
- Pipelines team has adopted Parquet/Dask as a preferred analysis format.

Technical progress

- Adjacent to the project, there are several efforts at understanding astronomical use cases in these frameworks. This is an active research topic, and "best practices" are not yet firmly established. Using it for ZTF at UW.

AXS: A framework for fast astronomical data processing based on Apache Spark

Petar Zečević,[1,2] Colin T. Slater,[3] Mario Jurić,[3] Andrew J. Connolly,[3] and Sven Lončarić[4]

[1] *Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia*
[2] *Visiting Fellow, DIRAC Institute, University of Washington, Seattle, USA*
[3] *DIRAC Institute and the Department of Astronomy, University of Washington, Seattle, USA*
[4] *University of Zagreb, Croatia*

Broader Policy Questions

- We don't want to advertise a service that is unusable due to resource constraints.
- If a single query takes up a user's entire monthly quota, that's bad.
- Jupyter is already going to occupy most, if not all, of the 10% share for user processing.

Broader Policy Questions

- We would be much safer if we could adopt a "bring your own compute" model.
- We would provide a service with enough resources for users to try it out, and users could purchase their own resources when necessary.
- Several top-level policy questions involved, particularly data rights and support for non-LDF DAC's and commercial providers.

Summary

- We have a suite of strategies for supporting high-throughput computation.
- For the most intensive and challenging use cases, we have a conceptual plan for how to technically support beyond-SQL processing frameworks. Some prototype pieces in place, and lots of flexibility available.
- Resourcing and policy choices (at project and agency levels) are the greatest uncertainties; implementation must follow from that.