LSST + Amazon Web Services Proof of Concept

Kian-Tat Lim, Leanne Guy, and Hsin-Fang Chiang

2019-03-13

1 Goals

We would like to verify that a cloud deployment of the LSST Data Management (DM) Data Release Production (DRP) is feasible, measure its performance, determine its final discounted cost, and investigate more-native cloud options for handling system components that may be costly to develop or maintain.

2 Data Release Production

The Data Release Production involves several components.

2.1 Datasets

Various sizes are available, from ci_hsc (8 GB) to HSC Public Data Release 1 (80 TB) to DESC DC2 (1300 TB).

The datasets could be hosted on various types of storage, ranging from a shared POSIX-compliant filesystem to an object store, although our code currently supports only the shared filesystem model.

2.2 Pipelines

We would want to run the PoC with representative algorithms performing instrument signature removal (ISR), image calibration, measurement, coaddition, coadd measurement, and forced photometry.

If certain pipelines are not yet ready, they can be mocked up or dropped from the processing (if at the tail).

Pipelines generally consume 1 to 4 GB of memory per core, but some tasks may consume considerably more (perhaps 40 GB), and the initial DRP demonstration workflow using the ci_hsc dataset took about 3 core-hours to run. We would expect the CPU consumption to scale approximately linearly with the dataset size.

2.3 Middleware

We should use the newest version of our processing pipeline tasks using the Generation 3 Middleware (”Gen3”). This comprises:

- an I/O layer called the Data Butler which includes a database back-end called the Registry that tracks datasets, both read and created,
- a command-line utility to execute pipeline tasks that queries the Registry to determine what inputs and outputs are needed,
- and a workflow system based on Pegasus and HTCondor that accepts a definition for a set of pipelines to execute and the data to execute them on (called a ”campaign”), generates a graph of all of the processing needed, and manages that processing on a cluster of batch nodes.

The Gen3 Butler Registry is expected to require 3 inserts in one transaction for each dataset produced by each execution of each pipeline task for a total of about 7000 inserts for the ci_hsc dataset processing. In addition, SELECT queries are issued by the workflow graph generator at a rate of a few per task execution for a total of around 500 SELECTs for the ci_hsc dataset processing. Pipeline tasks are expected to execute at a typical rate of one per core-minute, but this is highly variable (ranging from a few seconds to tens of minutes). The current registry implementation uses SQLite through SQLAlchemy; a more robust implementation will be needed to scale up to large processing jobs.

3 Proposed Phases

At each phase, we would observe wall-clock time, compute efficiency, memory usage, and other performance parameters. We would test the system at various levels of scaling appropriate to the dataset size. Actual usage of paid resources would be monitored, and appropriate cost/scale relationships would be derived.

AWS staff will consult on how best to use and configure AWS services to achieve the goals. They will learn the characteristics of LSST data and workloads to enable them to better estimate potential costs and determine available discounts for LSST use of AWS services. If needed, AWS staff can modify the LSST code base (which is all Open Source) to make it more AWS-compatible.

3.1 Phase 1: (0.5 month)

We would start by executing the Feb. 1 Gen3 demonstration on the cloud in a way that is as similar as possible to how we ran it at NCSA. The simplest way to start might be to use pre-allocated EC2 nodes as batch workers for HTCondor. We would use the ci_hsc dataset and run with a shared filesystem at a small scale.

3.2 Phase 2: (0.5 month)

Move HTCondor to an AWS-specific configuration for compute using the most cost-efficient compute resources. This may involve changing the HTCondor back-end to use Kubernetes and SmartFleet.

3.3 Phase 3: (1 month)

Move the Butler Registry onto a cloud-based SQL RDBMS (perhaps Aurora or PostgreSQL).

3.4 Phase 4: (1 month)

Move the datasets into S3. Investigate moving Pegasus, HTCondor, and the Butler to an AWS-specific configuration that would provide transparent-to-the-pipelines access to datasets on S3.

3.5 Phase 5: (1.5 months)

Scale up to larger datasets. HSC PDR1 is the prime target for a demonstration at the LSST 2019 Project and Community Workshop from August 12–16, with DESC DC2 as a stretch goal, likely after the August timeframe.

If the Butler Registry proves to be a bottleneck, investigate alternatives. These include higher-performance non-SQL databases like DynamoDB or ElastiCache, or shared-nothing/reingest mechanisms that would minimize the number of operations on the Registry.

3.6 Phase 6 (optional):

Investigate monitoring, control (e.g. terminating and restarting some processing), and other operational capabilities.

3.7 Phase 7 (optional):

Investigate how LSST staff, collaboration members, and science users could use their own AWS accounts to execute DRP-like processing with data-rights-controlled access to LSST datasets stored on S3.

3.8 Phase 8 (optional):

Investigate other possibilities for executing workflows using AWS-native tooling.

4 Reports and Conclusion

Reports of each phase executed, including what was accomplished, measurements made, and lessons learned, will be prepared by LSST and AWS staff. Given the cost/scale relationships determined by the PoC, fully discounted cost projections for executing the DRP on AWS would be generated by AWS for comparison with LSST TCO estimates.

5 Community Broker

LSST will produce an alert stream comprised of approximately 10 million alert packets per night [DMTN-102]. Each alert packet will contain about 80 kilobytes of measurements of an object that has been observed to change in an image. (The Zwicky Transient Factory https://www.ztf.caltech.edu/ is a current project that is producing a stream about 10% of the size of LSST's.) Recently LSST has issued a call for letters of intent for community brokers [LDM-612; LDM-682] that can receive and add value to this stream and distribute it to users. AWS may be interested in engaging in collaborations with existing brokers to provide technical expertise or to use AWS-specific services to receive, process, and distribute the alert stream. While setting up such a broker is not part of the PoC, LSST can facilitate discussions between AWS and potential partners.

A References

References

[LDM-612], Bellm, E., co authors, 2018, *Plans and Policies for LSST Alert Distribution*, LDM-612, URL `https://ls.st/LDM-612`

[LDM-682], Bellm, E., co authors, 2019, *Call for Letters of Intent for Community Alert Brokers*, LDM-682, URL `https://ls.st/LDM-682`

[DMTN-102], Graham, M., Bellm, E., Guy, L., et al., 2019, *LSST Alerts: Key Numbers*, DMTN-102, URL `https://dmtn-102.lsst.io`,
LSST Data Management Technical Note