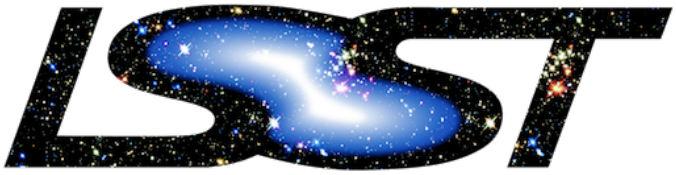

LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Potential proofs of concept for cloud deployment

William O'Mullane and John Swinbank and K.T. Lim and Margaret Gelman and Xiuqin Wu and Fritz Mueller

DMTN-078

Latest Revision: 2018-08-27

1 Introduction

In this brief note we wish to discuss some potential routes to collaboration with Google. We will revisit some of the potential ways forward to the cloud from DMTN-072 and try to quantify what a Proof of Concept might look like. There is also scope for technical collaboration on infrastructure.

If we go ahead with one specific POC we should define more clearly the goal and duration of it and how that might feed into future operations.

2 Technical collaboration

LSST is already a heavy user of Kubernetes (K8s) both internally and via Google Cloud. We have experienced specific technical issues running on our own K8s enabled hardware such as excessive start-up times[1] and security concerns around integration with the GPFS filesystem. We believe we are pushing the boundaries of deployments of this sort: any help would be appreciated, and we provide a testbed outside Google for this type of work.

3 Cloud proofs of concept

Much of the LSST Data Management System (DMS) — and, certainly, everything described below — is deployable using Kubernetes. This provides us with a lot of flexibility to port our system across service offerings, and would enable us to easily adopt a hybrid cloud plus on-premises infrastructure.

Moving to a cloud-based infrastructure could potentially save on personnel, as no hands-on hardware maintenance would be required. Although this is equivalent to a relatively small fraction of the construction budget, it would represent a substantial sum dedicated to non-core-business during operations.

We can probably not move wholesale to the cloud: we are committed to providing a physical Data Access Center (DAC) in Chile, and some physical hardware must remain on the mountain and in the Commissioning Cluster. However, there are potentially a number of opportunities to migrate a subset of DM services to the cloud if we could see a sensible way forward.

[1] Solved for now.

3.1 Qserv: the LSST database

The bespoke Qserv database system [LDM-135], under development at SLAC, has not yet been tested in a cloud based environment. However it is now deployable with Kubernetes, and no longer requires special hardware: physically attached storage is needed, but this is available on cloud offerings.

Qserv's performance characteristics are well understood, and would form an excellent testbed for LSST operations in a cloud environment. However, Qserv itself is primarily useful in the context of the Science Platform (Section 3.2), so the longer term benefits of a Qserv-only deployment would be limited.

3.1.1 Potential needs

To set up a Qserv instance we would need at least 40 large nodes with physically attached storage on the order of 5 terabytes per node. To run a set of convincing tests we would need that up for order two months.

Qserv workers currently use 2x8 cores, 128GB memory, 10x2TB local-attached disks, so google n1-highmem-16 might be a good fit. The local-attached disks and large number of spindles are primarily designed for high bandwidth sequential reads; it's possible that network-attached disk can provide similar rates. We'd like aggregate read bandwidths across the cluster of >100 GB/sec. IOPS are not really a concern; around 1000 IOPS aggregate across the cluster is desirable, which should be no problem given the number of nodes and (real or virtual) spindles.

We also need a "czar" node that has higher performance: 3 TB of SSD (2 TB persistent, 1 TB could be temporary), 6 TB of disk storage, and a large amount of memory. n1-highmem-64 probably fits.

This would be a demonstration only — the final catalogs (2032) will be order 15PB and the number of nodes and attached storage eventually have to scale to that size.

3.2 Cloud-based Science Platform

The LSST Science Platform [LSE-319] envisions three key ways in which scientists will interact with LSST data: through a visualization portal, a Jupyter notebook-based interface and through a variety of web services. These serve as an interface to images stored on disk and catalog data stored in the Qserv system (Section 3.1). This is an intrinsically cloud-oriented

approach to the problem of accessing large volumes of LSST data: it is based upon user code being co-located with the data on which it is running.

A key benefit of a cloud-based Science Platform would be scalability: when user demands exceed the 10% of the compute budget dedicated to serving them, more capacity would at least be available even if it had to be purchased on demand. There is no analogue to this in terms of on-premises infrastructure, as cloud bursting from our internal cloud infrastructure to a commercial provider would require transferring potentially large amounts of data.

3.2.1 Aside: Public Data Releases

LSST data becomes public two years after its initial release. However there is no project budget allocated to serve this public data, although it remains scientifically valuable. One imagines that a public, cloud-based version of the Science Platform serving old data could be a valuable resource e.g. for enabling science in underdeveloped countries.

3.2.2 Potential needs

All the Science Platform components are deployable with Kubernetes. The Qserv database component is a fixed size resource as discussed in Section 3.1.1. In addition one or preferably two servers should be provisioned for the web services.

Alongside that one needs to have the JupyterHub environment [2]; depending on the assumed load, this is relatively modest as it requires only ~ 2 servers to set up, and it is recommended to have 2 CPUs per simultaneous user. For a proof of concept let's assume we would go with 20 simultaneous users to 40 CPUs or 10 nodes depending on the type of node. Each user should also have around 4GB of RAM. Ideally, we would also have a batch system controlling additional compute resources for more extensive analyses, but perhaps for a proof of concept this may be treated as a desire rather than a requirement.

Firefly also requires at least a pair of servers — these should be 32 cores with 128 GB memory. In addition these should have a shared disk volume of order 500GB, preferably SSD.

Finally there is a filesystem to store the image data. Our current code assumes a POSIX filesystem, but we have made some modifications towards supporting an object-store back end. Additional investment in this direction is unlikely to happen before Fall 2018. LSST will pro-

[2] see https://github.com/lsst-sqre/jupyterlabdemo

duce over its lifetime around 60 PB of raw image data, the final data volume including the processed images is estimated to be around 0.5 EB. For the POC 1 PB would be sufficient to evaluate performance and management of a large disk volume.

For the proof of concept we could leave out the Prompt Products Database (PPDB this is a conventional e.g. Oracle database)[3].

3.3 Cloud based prompt processing

Prompt processing [LDM-151] is the umbrella term used to describe processing which produces data products continuously while LSST operates. Broadly, this falls into three categories:

1. *Image reduction and differencing*, in which images are received from the camera, calibrated, and compared to deep template images of the same part of sky to identify transient and variable sources;
2. *Alert distribution*, in which notifications of transients and variables are distributed to the community;
3. *Moving objects processing*, in which solar system objects are identified and their orbits tracked.

We expect to issue around 10^7 alerts per night during normal operations. Further, the project is required to make these alerts available to the community within no more than 60 s of the telescope shutter closing. This imposes stringent latency and throughput requirements on items 1 & 2 above. Moving object processing can be run during the day, and is therefore a less challenging — and for this purpose less interesting — use case.

One night of LSST observing generates approximately 20 TB of data in a "bursty" fashion (we visit each field for a total of 37 s, taking two consecutive exposures which are combined by the data processing system). In addition, the compute resources are more lightly loaded during the daytime, when they are used to perform fewer and simpler analyses of calibration images. In order to meet our latency requirements, we have invested in fast networking to enable rapid transfer of this data to processing systems at NCSA.

[3] This PPDB would be required for the Prompt Processing POC if we go that way

Once on the compute systems, data from each of the 189 CCDs in the camera is processed in parallel: broadly, we expect a single CPU with access to 4 GB RAM to handle each CCD independently, taking somewhat less than a minute to complete. In operations, we anticipate deploying two separate clusters, or "chains", with each processing alternating visits.

These processing chains will then feed the results of their processing to the alert distribution system, based on Apache Kafka[4], for distribution to the community. A prototype of this latter system has already been deployed on Amazon AWS DMTN-028.

This suggests two, related, proof of concept exercises:

- Demonstrate rapid ingest and image processing;
- Demonstrate at-scale alert distribution.

3.3.1 Potential needs

Demonstrating image processing at scale could be achieved with a single processing chain (ie, 189 CPU cores, with access to 4 GB RAM per core - 189 xn1-standard-64i would probably suit). Around 60 TB disk storage is required for a single night of data (including processed data products). However, a smaller dataset could be defined for testing purposes.

Prompt image processing would also require a database instance to serve as the Prompt Products Database (PPDB). This requires a relatively high-performance database as it is part of our near-realtime processing. It eventually needs to store about 80 TB of data, plus indexes, but for testing we can probably use 10-20 TB. While it has been simplest to run this on a single node to date, one of the things we'd like to investigate is the possibility of sharding it (by rows; it is not possible to shard this data by columns). On a single node, we believe the storage will have to be on SSD to provide sufficient IOPS; we likely need tens of thousands of IOPS here. Our prototypes have used up to 32 cores and 512GB memory.

Deploying a realistic alert distribution system would require three systems, each with access to 24 CPU cores and 80 GB RAM.

[4] http://kafka.apache.org

4 Conclusion

A number of potential POCs are discussed above with approximates sizes/needs. We should pick one or more to develop further.

A References

[1] **[LDM-135]**, Becla, J., Wang, D., Monkewitz, S., et al., 2017, *Data Management Database Design*, LDM-135, URL https://ls.st/LDM-135

[2] **[LSE-319]**, Jurić, M., Ciardi, D., Dubois-Felsmann, G., 2017, *LSST Science Platform Vision Document*, LSE-319, URL https://ls.st/LSE-319

[3] **[DMTN-072]**, O'Mullane, W., Swinbank, J., 2018, *Cloud technical assesment*, DMTN-072, URL https://dmtn-072.lsst.io,
LSST Data Management Technical Note

[4] **[DMTN-028]**, Patterson, M.T., 2018, *Benchmarking a distribution system for LSST alerts*, DMTN-028, URL https://dmtn-028.lsst.io,
LSST Data Management Technical Note

[5] **[LDM-151]**, Swinbank, J.D., et al., 2017, *Data Management Science Pipelines Design*, LDM-151, URL https://ls.st/LDM-151

B Acronyms

The following is a complete list of acronyms used in this document.

Acronym	Description
CCD	Charge-Coupled Device
CPU	Central Processing Unit
DAC	Data Access Center
DM	Data Management
DMS	Data Management Sub-system

DMTN	DM Technical Note
EB	Eclipsing Binary
GB	GigaByte
GPFS	General Parallel File System
LDM	Light Data Management
LSE	LSST Systems Engineering (Document Handle)
LSST	Large Synoptic Survey Telescope
NCSA	National Center for Supercomputing Applications
PB	PetaByte
POC	Proof Of Concept
POSIX	Portable Operating System Interface
RAM	Random Access Memory
SLAC	Stanford Linear Accelerator Center
SSD	Solid-State Disk
TB	TeraByte
s	second; SI unit of time

C More focused PoC for Google cloud

this was originally edited as a google doc and is put here as a matter of record.
We feel the proofs of concept (PoCs) outlined in DMTN-078 that provide the best combination of utility for LSST's decision-making and readiness to be implemented are one based on Qserv (section 3.1) and one based on Prompt Processing (section 3.3). If time remains, a demonstration of "cloudbursting" LSST Science Platform processing from the LSST Data Facility's private cloud to the Google Cloud (a subset of 3.2) would be useful as well.

Assuming that we will be starting approximately 2018-09-01, Qserv will have been fully converted to Kubernetes deployment, including the internal replication management system. Data from the WISE survey is already available to be loaded into Qserv, as is a synthetic dataset used for performance benchmarking that is approximately 30% of the first-year LSST results. We expect that at least two large tables (Object and Source) from a large reprocessing of Subaru Hyper-SuprimeCam (HSC) data should also be available by then for loading in "DPDD-like" form (the form of the data specified by the LSST Data Products Definition Document).

Proposed Qserv PoC execution plan, with each step incorporating analysis and debugging of

any functional or performance issues discovered:

1. Deploy Qserv on Google Cloud with local-attached disks.
2. Load KPM30, WISE, and HSC datasets into Qserv.
3. Execute queries from DMTR-52 and DMTR-16 against datasets.
4. Adjust deployment to use network-attached disks.
5. Re-execute queries against datasets.
6. Select one or more Google database services (Cloud SQL, Datastore, Bigtable, etc.) that best matches data and queries; load data into the service.
7. Re-execute queries (if possible) against database service.
8. Analyze the potential for solving any query supportability issues (e.g. spherical geometry, near-neighbor queries, simultaneous users).
9. Analyze the potential for "next-to-database" processing (DMTN-086) with data in Google database service.

The Prompt Processing PoC could overlap somewhat with Qserv if resources are available, but otherwise we would anticipate it starting some time before the end of the calendar year. This PoC will require more shimming and potential technology replacement than the Qserv one. For example, our systems currently make use of POSIX filesystems. By the time of the PoC, we might have written plugins for our I/O client that can access an object store.

Proposed Prompt Processing PoC execution plan:

1. Perform high-bandwidth transmission of CCD images from machines at NCSA (e.g. in the Level 1 Test Stand) to Google Cloud storage.
2. Stand up an instance of the Prompt Products Database. This currently is anticipated to run in Oracle; we could attempt to substitute a different technology here.
3. Load master calibration images and templates into Google Cloud storage.
4. Deploy an instance of the Alert Distribution and Filtering service to Google Cloud.
5. Deploy a set of simulated alert clients both within and outside Google Cloud.

6. Execute Alert Production in batch mode on Google Cloud machines, sending alerts to the distribution service.
7. Investigate the possibility of sharding the Prompt Products Database.
8. Analyze the applicability of Google workflow/orchestration services (Cloud Composer, Cloud Dataflow) for executing Alert Production.

The LSST Science Platform is already deployable on Google Cloud, but only with its own internal user file space that is not shared with any other deployment, and with access to LSST data products only through potentially-slower Internet-exposed Web services.

Proposed LSP Cloudburst PoC execution plan:

1. Determine an appropriate configuration to allow JupyterHub at NCSA (and Chile ?) to create pods both locally and remotely in Google Cloud.
2. Determine how authentication and authorization can be synchronized between NCSA and Google Cloud.
3. Determine how user files and databases can be shared between NCSA and Google Cloud. VOSpace and/or WebDAV are potential mechanisms here.
4. Determine whether LSST data products need to be permanently resident in Google Cloud or if Web APIs are sufficient.