SLAC-PUB-12962
November 2007

Astronomical Data Analysis Software and Systems XVII **O5.4**
ASP Conference Series, Vol. XXX, 2008
J. Lewis, R. Argyle, P. Bunclarck, D. Evans, and E. Gonzales-Solares, eds.

Organizing the Extremely Large LSST Database for Real-Time Astronomical Processing

Jacek Becla[1], Kian-Tat Lim[1], Serge Monkewitz[2], Maria Nieto-Santisteban[3], Ani Thakar[3]

Abstract. The Large Synoptic Survey Telescope (LSST) will catalog billions of astronomical objects and trillions of sources, all of which will be stored and managed by a database management system. One of the main challenges is real-time alert generation. To generate alerts, up to 100K new difference detections have to be cross-correlated with the huge historical catalogs, and then further processed to prune false alerts. This paper explains the challenges, the implementation of the LSST Association Pipeline and the database organization strategies we are planning to use to meet the real-time requirements, including data partitioning, parallelization, and pre-loading.

1. Introduction

The Large Synoptic Survey Telescope is a proposed ground-based 8.4 meter telescope with a 3.2 gigapixel camera. Once in production in 2014, it will produce a new image every 15 seconds, leading to a 55 petabyte raw image archive 10 years later. The survey is expected to catalog 50 billion stars and galaxies based on over 3 trillion individual astronomical source detections. Among its main challenges is real-time transient alert generation (Becla et al. 2006). To generate an alert the pixel data must be reduced and cross-correlated with the existing catalogs, all in under a minute.

2. Alert Generation

It is expected that LSST will generate some 100 thousand alerts per night. To generate alerts, each pair of back-to-back exposures, called a *visit*, must be processed. The processing will occur at *Base Camp*, a computing center near sea level below the mountaintop telescope. To generate alerts for a given visit:

1. The two exposures must be transfered to the Base Camp.
2. The two exposures must be processed by the Image Processing Pipeline.
3. New detections must be generated by differencing with a template image.
4. The new detections must be associated with existing astronomical objects.
5. A decision has to be made which detections should trigger alerts.

[1]Stanford Linear Accelerator Center, Menlo Park, CA, USA

[2]California Institute of Technology, Pasadena, CA, USA

[3]The Johns Hopkins University, Baltimore, MD, USA

Contributed to 17th Annual Astronomical Data Analysis Software And Systems Conference (ADASS 2007) , 9/23/2007-9/26/2007, London, UK

Work supported in part by US Department of Energy contract DE-AC02-76SF00515

The process of associating new detections with existing astronomical objects (4^{th} step above) is run as a separate LSST pipeline, the *Association Pipeline*, and it involves the following tasks:

1. New detections (*difference detections*) are cross-correlated with the existing Object Catalog. The correlation involves a spatial search within a 0.05 arcsec radius. Matching detections are marked.
2. The unmatched detections are cross-correlated with the Moving Object Catalog, which contains information about known moving objects and their predicted positions at the time when the exposures were taken. Matching detections are again marked.
3. Entries in the Object Catalog corresponding to matching detections are updated. For each remaining unmatched detection, a new entry is created.

The difference detections are then passed to the *Alert Generation Pipeline* which examines in detail the detections and their corresponding objects or moving objects if they exist and decides on their alertability.

All updates to the Object Catalog must be available when subsequent observations are processed the same night.

It is expected LSST will deal with an average of 40 thousand new detections and 4 million historical objects per visit, with peaks of 100 thousand and 10 million respectively. New objects per visit should be around 1 thousand.

3. Design Details

It is easy to notice the whole process could be very disk I/O intensive. The main contributors include (a) locating and reading 4–10 million objects corresponding to the observed field of view (FOV) out of the 50 billion row / 100 terabyte Object Catalog, and (b) updating 40–100 thousand rows in the Object Catalog. "a" implies access to a potentially large volume of data, and "b" implies many small, sparse updates and writes.

3.1. Minimizing I/O

In order to minimize disk I/O, the cross-match should be done only against the objects in or near the processed FOV. If we further sequentialize (derandomize) I/O by clustering data accessed together, that reduces the problem from 100,000 × 50 billion to 100,000 × 4 million, or from 100 terabytes to 7 gigabytes. In practice, it implies we need to store together objects that are close in the spatial coordinate system.

Further, the I/O can be reduced by fetching only the columns used during cross-match. Each row in the Object Catalog consists of close to 300 columns occupying 1,658 bytes, while the spatial cross-match needs access to only 9 columns: `objectId`, `ra`, `decl`, and a variability probability for each filter. That reduces the problem from 7 gigabytes down to 130 megabytes. In row-oriented database systems, entire rows are stored together on disk. To avoid sparse reading (skipping unwanted columns) a specialized copy with the 3 columns will be maintained.

Further speed up can be achieved by partitioning the data in round-robin fashion across several disks. We chose to partition data such that for any given FOV the corresponding data can be fetched by reading a small number of partitions. Using spreadsheet-based analysis and some testing we determined a good

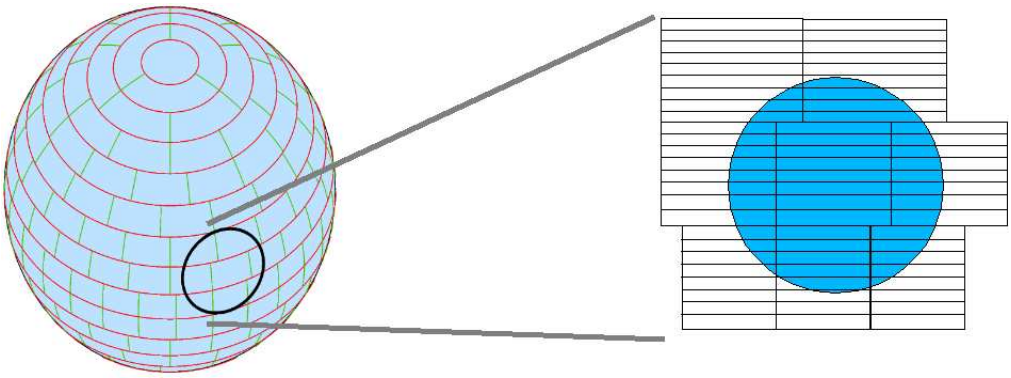

Figure 1. Globe partitioning

LSST partitioning scheme: the Object Catalog should be divided into *declination stripes*, with each stripe partitioned in RA to form a total of 330,000 chunks. Stripes should be a fraction of a FOV high to minimize wasted I/O around the circular FOV. The partitioning scheme is shown in Figure 1.

It is worth noting that the optimizations outlined above lead to a compact FOV data set which easily fits in memory of one of today's commodity servers.

The LSST Online Control System will know at least 30 seconds in advance the coordinates of the *next* FOV. That allows further optimizations: data for the next FOV can be prepared before the time-critical processing is started. To take advantage of that, we split the Association Pipeline into 3 distinct phases:

1. pre-process: data is fetched from disk to memory, reorganized and indexed
2. real-time: in-memory snapshot of the data is used, that includes reading and updating existing data, and writing new data
3. post-process: the changes are written to disk. If some data is needed for the subsequent FOV, it will be kept in memory.

To avoid losing data, we expect to fully mirror the environment and provide hot-swap capabilities.

As explained earlier, the Alert Generation Pipeline will need to examine in detail the potentially alertable detections and corresponding objects, if they exist. In almost all cases the alertable detections will correspond to variable objects. Since a very small fraction of all objects are variable (less than 5%), it makes it easy to fit all variable objects for a given FOV in memory.

All the above optimizations lead to an architecture where there is almost no disk I/O activity during the time-critical period of the Association Pipeline. The only remaining disk I/O is triggered by fetching objects which used to be classified as static objects that are now being reclassified as variable objects. In practice, there will be less than 1,000 such objects per visit, generating less than 2 megabytes of disk traffic (1,000 $\times$ 1,658).

In order to keep up with the data, each pre-process and post-process may not take longer than the visit duration (30 seconds). In such an environment, during steady state there should not be more than one pre-process, one real-time and one post-process running at any given time. That is easily doable on a modest present-day machine with a disk array capable of delivering $O(100)$ MB/sec.

3.2. Maximizing Computational Efficiency

In order to further speed up the association process, we optimize the in-memory snapshot of the data by dynamically dividing each stripe into *declination zones* (Szalay et al. 2004), as shown in Figure 1. Although the optimal zone height for cross-matching is equal to a single search radius, we use zones that are several search radii high to minimize index overhead. Furthermore, we pre-sort data within zones by RA. During the time-critical period the actual match is done by matching detections and objects within zones, which can be trivially parallelized.

We also determined that it is faster to push the computation from the database to the application. In the application we can take advantage of highly-specialized algorithms to do the computation faster than a typical database does. Such an approach involves moving data to computation, which is somewhat counter-intuitive. It works in our environment because we first significantly reduced the size of the working data set.

4. Results

We ran a set of tests for the average real-LSST scale: 40 thousand detections and 4 million objects per visit, using a single 2002-era SunFire V240 UltraSparc IIIi 1.5 GHz server.

As explained above, the prepare phase must finish under 30 seconds. According to the official requirements the time-critical cross-match must finish in 10 seconds.

In the tests we ran, the prepare phase took 9.4 sec, and the real-time phase took a little more than 1 sec: reading and indexing new detections 0.91 sec and cross-match 0.14 sec.

The testing involved spatial cross-match only. Examining the full contents of variable objects (part of Alert Generation) was outside the scope of this year's data challenge and was not tested. Given the optimizations planned, namely keeping the bulk of the data in memory and only fetching 2 megabytes of data from disk, it is expected this will take 1–2 seconds.

5. Conclusions

One of the important LSST missions is to provide real-time alerts triggered by transient objects. From the database perspective that implies some 40,000 detections must be rapidly matched against some 4 million entries that have to be extracted from the historical 50-billion-row Object Catalog. We proved that the average LSST-scale spatial cross match of the detections with existing objects can be achieved on a single, modest server. The optimizations include organizing the data efficiently, replicating, prefetching the working data set to memory and pushing the computation to the application.

References

Becla J. et al. 2006, in Proc. SPIE Vol. 6270, 62700R
Szalay, A.S. et al. 2004, MSR-TR-2004-32