VERA C. RUBIN OBSERVATORY

APDB Update

Fritz Mueller / Andy Salnikov
SLAC National Accelerator Laboratory
Vera C. Rubin Observatory Data Management

NSF NOIR Lab AURA SLAC U.S. DEPARTMENT OF ENERGY

Background

- The Alert Production Database (APDB) is an internal system component which must support Alert Production at each visit by:
  - Answering queries for all known difference-image-objects within the visit FOV
  - Answering time-series queries for all such objects that match detections in the visit
  - Accepting updates to all matched objects
  - Accepting inserts of new detections and new forced photometry
- The data density and available time budget (~10 seconds) are aggressive.
- A framework to simulate APDB query load has been developed and put to use to evaluate potential implementations.
- Over the past year, a prototype implementation based on Apache Cassandra has been developed and tested at scale. Recent results summarized here, detailed in Jira (links follow), and a forthcoming DMTN.

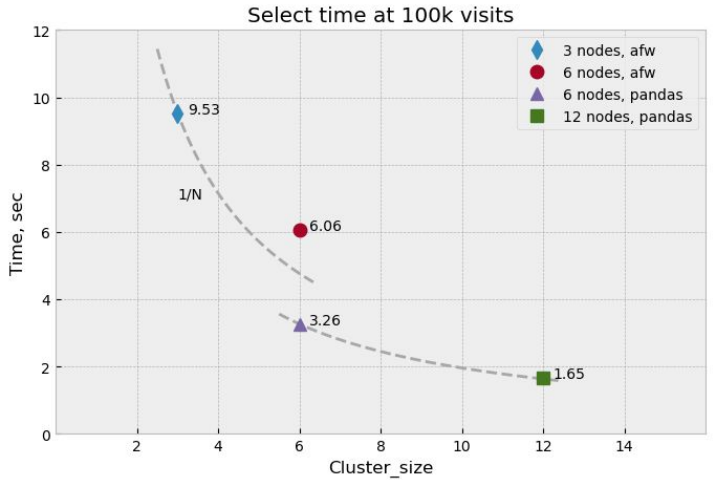
GCP Test Configuration

- Google Cloud Platform tests builds upon earlier tests with PDAC
  - Summarized in DMTN-156
- Few things we learned with PDAC:
  - Low-latency, high IOPS storage is critical for APDB -- means NVMe SSD
  - Estimate of the data sizes, including multiple replicas
  - Helped us with planning of GCP tests (DMTN-162)
- Main motivation for GCP testing is to prove that we can scale performance horizontally, and demonstrate operation at target steady-state of 1yr. of history
- Try different options and optimizations
- Many details in DM-27785 epic

- Setup:
  - Server-side: 3, 6, or 12 nodes; n2, 32 vCPU, 64 GiB; locally attached NVMe disks 8x375 GiB; Ubuntu
  - Client-side: 6 nodes; e2, 32 vCPU, 16 GiB; CentOS7 LSST image; 189 MPI jobs

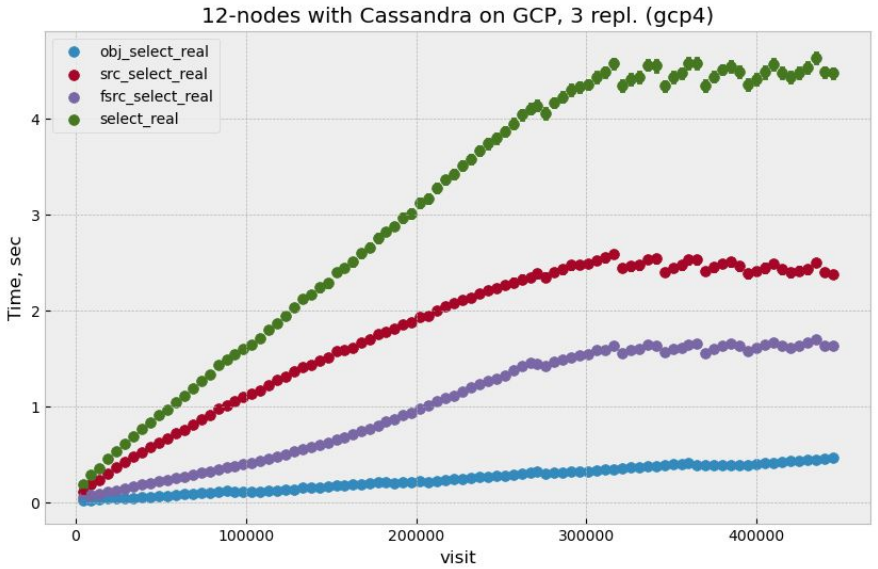
Scalability Test

- Ideally performance should scale linearly with the number of nodes in cluster
- Run with 3, 6, 12 nodes and compare
- Select time is our main metric (and bottleneck)
- 3-node result is consistent with PDAC
- 6-node did not result in expected time reduction
  - A lot of CPU time on client side for afw.table conversion
  - Replaced afw with pandas gives significant boost
  - Pandas is current AP-preferred format for SQL backend
- With pandas going from 6 nodes to 12 reduces select time in half
  - pandas is still using significant CPU, more on that later



Year Long Test

- Select time grows with the amount of data fetched from (forced) sources
- AP needs 12-month history of sources
- Expect select time grow linearly and plateau after 12 month, but have to demonstrate it as well
- Generated 450k visits, corresponds to approx. 1.5 years of data (took 2 weeks)
- With 12-node cluster select time plateaus at ~4.5 seconds
- Select time for DiaObject continues to grow slowly, need to understand what we can do to limit that growth



Test Partitioning Options

- APDB data in Cassandra is partitioned
  - MQ3C(10) spatial partitioning; one month time partitioning for sources
  - Balance between number of queries and amount of data returned
  - Time partitioning will likely be done via separate tables, for cost reasons
- Some freedom in formulating queries:
  - One large query “SELECT … WHERE partition IN (...)”
  - Many queries “SELECT … WHERE partition = ...”
  - Shifts processing from server side to client side
- Tested both options, did not see observable difference in select time
  - May indicate that performance is limited by server-side throughput
- Tested lower time partitioning granularity (2 months)
  - Some indication that it helps server side, but overall performance does not change
- Also tested Cassandra-native time partitioning vs. separate-table partitioning
  - No improvement observed
  - Separate-table partitioning complicates operations but can help with managing storage cost

Packing Data

- DiaSource/DiaObject tables are very wide
  - Database does not care about most of their columns, it only uses clustering/indexing ones
  - Packing all non-indexed columns into a BLOB could reduce server-side management overhead
  - But it can also increase storage size if packing format is very dynamic
  - Cassandra uses compression internally which could help
- Unpacking of BLOB is done on client side, more work for client
  - Not very fast if done at Python level, hope we can shift some of that to pandas
- Simple test was done to check if it can work
  - CBOR for serialization, compact binary format, but still packs column names
  - Tried to optimize conversion to pandas, not very successfully
  - Runs slower that non-packed case due conversion on client side
- Data size on disk increased by almost 100%
  - Practically rules out schema-less serialization (JSON, CBOR, etc.)
  - Might work with schema-based packing but implies schema management on client side
- May be an option with smarter serialization and faster conversion but needs significant development

Slow Pandas

- Things run significantly faster if we don’t convert results to pandas
  - Mismatch between database result which is a sequence of tuples and pandas internal storage
- One possibility for optimization, replace pandas with something that better matches result structure
  - Obviously need to coordinate with client-side requirements

High Availability

- Accidental “test”
- One of the server instances was mis-configured and became unresponsive for few hours
- Cluster continued to work and served requests from clients
- No noticeable performance degradation
- After fixing configuration the node restarted and all data in that replica recovered
  - Data recovery in Cassandra happens on read requests
  - Configurable consistency - reads and writes need quorum, 2 out of 3 replicas
- Demonstrates that cluster can operate efficiently with one node temporarily down

What Hasn't Been Tested

- DiaSources were only tested in write-only mode
  - No day-time re-association to SSObjects is implemented in ap_proto
  - Update operations are not trivial in Cassandra, may potentially affect reading performance
  - Should not be a big effect but would be useful to quantify
- There is a large uncertainty on the number of sources detected in a visit, may be worth to see how much slower we do with 50% higher numbers
- No concurrent reading from APDB by other clients
  - Unknown at this point what other clients might be and their access pattern
- Data management operations
  - E.g. move older data to slow storage to save cost or delete it entirely

Cost Analysis

- Sizing requirements
  - 12 nodes gives us reasonable performance, there is a chance we can optimize few things but we do not expect a lot of speed up, reducing cluster size to less than 10 nodes may be risky
  - Estimate for storage is 6 TiB per 100k visits for 3 replicas. If we could move data after 1 year to a slower storage we probably need space for 400-500k visits (24-30 TiB) on NVMe disks.
  - There is significant uncertainty in sources count which could translate into inflated numbers for storage and processing capacity
- Google cloud $$$ burn rate during tests
  - K-T estimated $5k/week when I ran that long one-year test
  - This is for the whole farm, 12 server nodes and 6 clients, client fraction should be small
  - Google estimates one server instance at $790/month, which is about $2,400/week for 12 instances, about half of what K-T saw
  - These are undiscounted prices (leftover POC credits; no “Rubin special”, no CUD)
- Rough procurement estimate for comparable 12-node cluster on-prem @ SLAC: ~$80K (plus data center operation costs)

Questions and Next Steps

- APDB API update
  - Cassandra backend has slightly different API due to partitioning
  - Plan is to introduce new API which will be implemented for both Cassandra and SQL (for local testing)
  - And potential pandas replacement
  - Need to discuss this with AP group to converge on something working
- AP developers have been working with small-scale sqlite implementation. What is needed next for developers in terms of scale, and by when?
  - For development it would be useful to have a small-size cluster available soon, small size can be a single host but fast storage is strongly preferred
- If AP devs need large scale, where would we set this up (Google, NCSA, USDF/SLAC?)
- Which upcoming milestones/rehearsals require at-scale APDB? (Same “where” question.)
- Can APDB data beyond 1yr. history simply be dropped, or must it be archived?