VERA C. RUBIN OBSERVATORY

APDB/PPDB Status Update

Andy Salnikov

DMLT meeting 2022-06-07

NSF NOIRLab AURA SLAC U.S. DEPARTMENT OF ENERGY

APDB & PPDB

APDB - Alert Production Database

- Output of AP pipeline - DIAObject, DIASource, DIAForcedSource
- Not including alerts
- Internal database, real-time updates, no external user access
- Optimized for AP performance, low latency queries, small set of queries

PPDB - Prompt Products Database

- Replicated from APDB
- Read and updated by Solar System Processing, possibly with some latency
- User-accessible, can run “arbitrary” queries (SQL is most suitable)
- Some updates need to be propagated back to APDB

APDB queries

For each visit and detector:

- Read the latest version of DIAObjects from detector region.
- Read 12 months of history of associated DIASources and DiaForcedSources.
- Save new or updated DIAObjects and their associated DIASources and DiaForcedSources.

Major concerns:

- Very tight time budget for the database interaction, few seconds is a reasonable goal. May be possible to pre-load data as soon as we know regions for the next visit, though this may interfere with writing in a currently processed visit.
- Performance and scalability are the key issues.
- High availability and transparent failover is critical for uninterrupted operations.

APDB tests with RDBMS

Initial APDB implementation used RDBMS.

- Tested with different backends and storage types at IN2P3, NCSA, and Google cloud (DMTN-113).

Main conclusions:

- Spinning disks are not adequate even for short time history.
- NVMe storage can provide reasonable performance for few months of data.
- For practical time scales we need a distributed scalable system, single host cannot handle the sort of load that we expect.

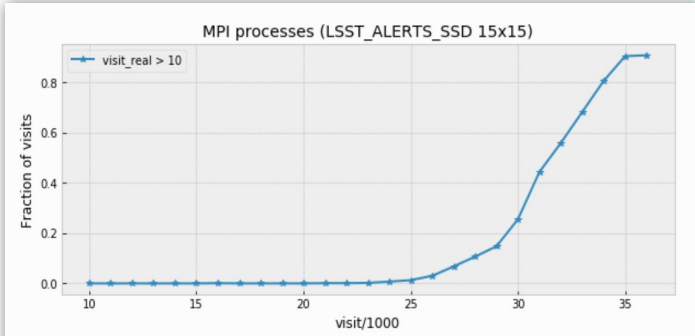


Figure 7 Fraction of visits with total total visit time higher than 10 seconds.

APDB tests with Cassandra

Cassandra was suggested as a scalable solution:

- NoSQL “database”, no support for arbitrary queries.
- Better control for data locality, how data is stored on disk.
- Partitioning (across nodes) and clustering (physical storage ordering) is used to optimize most critical queries.

Tests were performed with NCSA qserv cluster (DMTN-156) and Google Cloud at larger scale (DMTN-184):

- Reasonable performance with more than 12 months of data.
- Expected scaling behavior with the number of nodes in Cassandra cluster.
- Client-side performance (Pandas) can be improved.
- High availability features work reasonably without affecting performance.
- Estimated data size 2TB per 100k visits per replica.

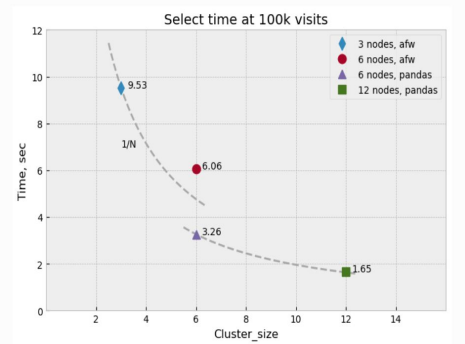


Figure 1 Summary of reading performance as a function of number of nodes. Performance is given as a total read time at 100k visits, with three-node and six-node `afw.Table` cases extrapolated to 100k. Curves represent ideal `1/N` scaling behavior.

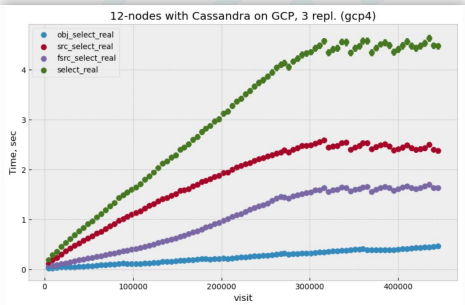


Figure 2 Time to read as a function of visit for all three tables, `select_real` is a sum of three other values. Total time plateaus after approximately 300k visits, small fluctuations are due to granularity of time partitioning.

Cassandra features

General features:

- Append-only updates for fast write operations with eventual compaction.
  - No separate UPDATE query, UPDATE=INSERT. DELETE=INSERT a tombstone.
- Reading can be slower because of that, best strategy is to avoid/reduce updates.
- Query language is SQL-like, but limited, e.g. no joins or foreign keys.

Partitioning for scalability:

- Distributing data across multiple nodes based on *partition key.*
- Partition key is the major restrictor for queries, in our case detector region.
- For source tables we also use time period (month) for partitioning.
- Partition size is critical for query performance and scaling.
- Whole partition is stored on disk sequentially, possibly in more than one file

Replication features

Replication:

- Each partition can be replicated to several nodes, critical for high availability.
- If one/few nodes become inaccessible operations will continue as long as consistency can be maintained.
- Automated repair when nodes come back.

Configurable consistency:

- Read/write operations do not need to wait for all replicas to respond
- Strong consistency can be configured by requiring half+1 replicas.
- Minimum number of replicas for strong consistency is 3.

APDB recent updates

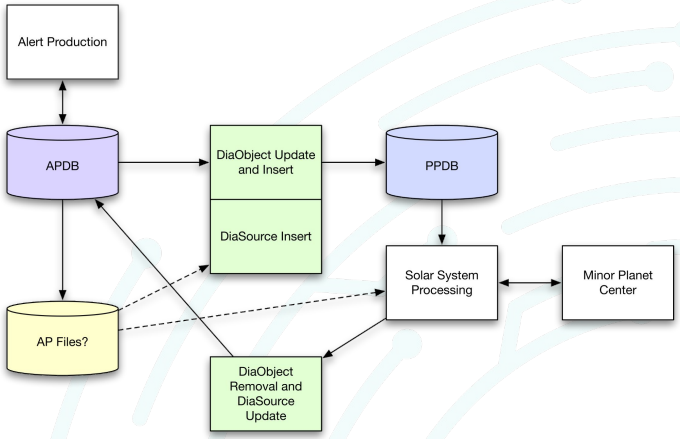
- Updated APDB API to work with both SQL and Cassandra implementations.
- Start preparations for integration of APDB with PPDB.
  - Added interface for reassigning DIASources to SSObjects.
- APDB schema has finally switched to felis format and moved to sdm_schemas.
  - Still uses APDB-internal felis parser, working on bringing felis into the stack.
  - May need to extend felis schema to support some Cassandra-specific features.
  - Some tables are de-normalized for more efficient queries (even in SQL implementation).
- AP pipeline is currently relying solely on SQL implementation.
- Would be nice to start testing AP with Cassandra at USDF.
  - No hardware at the moment.

PPDB ideas

- Not much development happened yet on DAX side.
- General idea is that we should copy APDB updates to PostgreSQL server.
- If single server capacity is not enough to serve user queries, we can consider adding one more PostgreSQL replica dedicated to user queries.
  - May also be useful for failover if main server experiences failure.
- Discussions about replication from APDB to PPDB (and back) are already happening.

APDB-PPDB Replication

- Replication from APDB to PPDB should happen regularly, some latency allowed.
- Reverse process can happen ~once a day.
- Needs a new set of tools to support replication.
- Queries covering large regions will be slow, could interfere with queries from AP.
- Need to think about an alternative way to send updates to PPDB:
  - e.g. save all updates as files on disk or separate Cassandra tables that are optimized for time-based selection.



Status

- Cassandra is a technology of choice for APDB.
  - Implementation is in a reasonable shape, waiting to be deployed.
- For PPDB the plan is to use PostgreSQL backend.
  - Separate replica may be needed for user/TAP queries.
- Replication between APDB and PPDB will need new development.
  - Some ideas already exist and need experimenting.

Links

Confluence

- 2021-09-21 PPDB Tag Up

TechNotes:

- DMTN-184: Testing Cassandra APDB implementation on GCP
- DMTN-162: Planning next round of APDB tests
- DMTN-156: Performance of Cassandra-based APDB implementation
- DMTN-113: Performance of RDBMS-based PPDB implementation (APDB)

JIRA:

- APDB-tagged issues