Experiences from Migrating from Oracle to PostgreSQL

Vinayak Hegde
&
Ravi Rayapati

Overview of Talk

- Definition of problem
- Migration strategy
- Exploration of alternatives / Tools
- Overview of features compatibility
- Conclusion

The problem space - The database

- The customer provides marketing automation to its clients (Data > 50TB with incr of 1TB/mth)
- The Oracle database
  - DB user data / campaign data
  - Needed campaign slicing / dicing capability
  - Fine-grained targeting
  - Video files were stored as LOBs
  - Unstructured data
- Imbalanced shards (tablespaces) Not based on size or had outgrown it over a period of time

The problem space - The environment

- Setup in own datacenter. Customer wanted to move out of the datacenter
- Single instance of database with a backup machine
- Used high-end hardware with Fusion IO as underlying storage.
- Problems with replication and backup (legacy).
- Archival was not thought about.
- Analytics was not extensive.

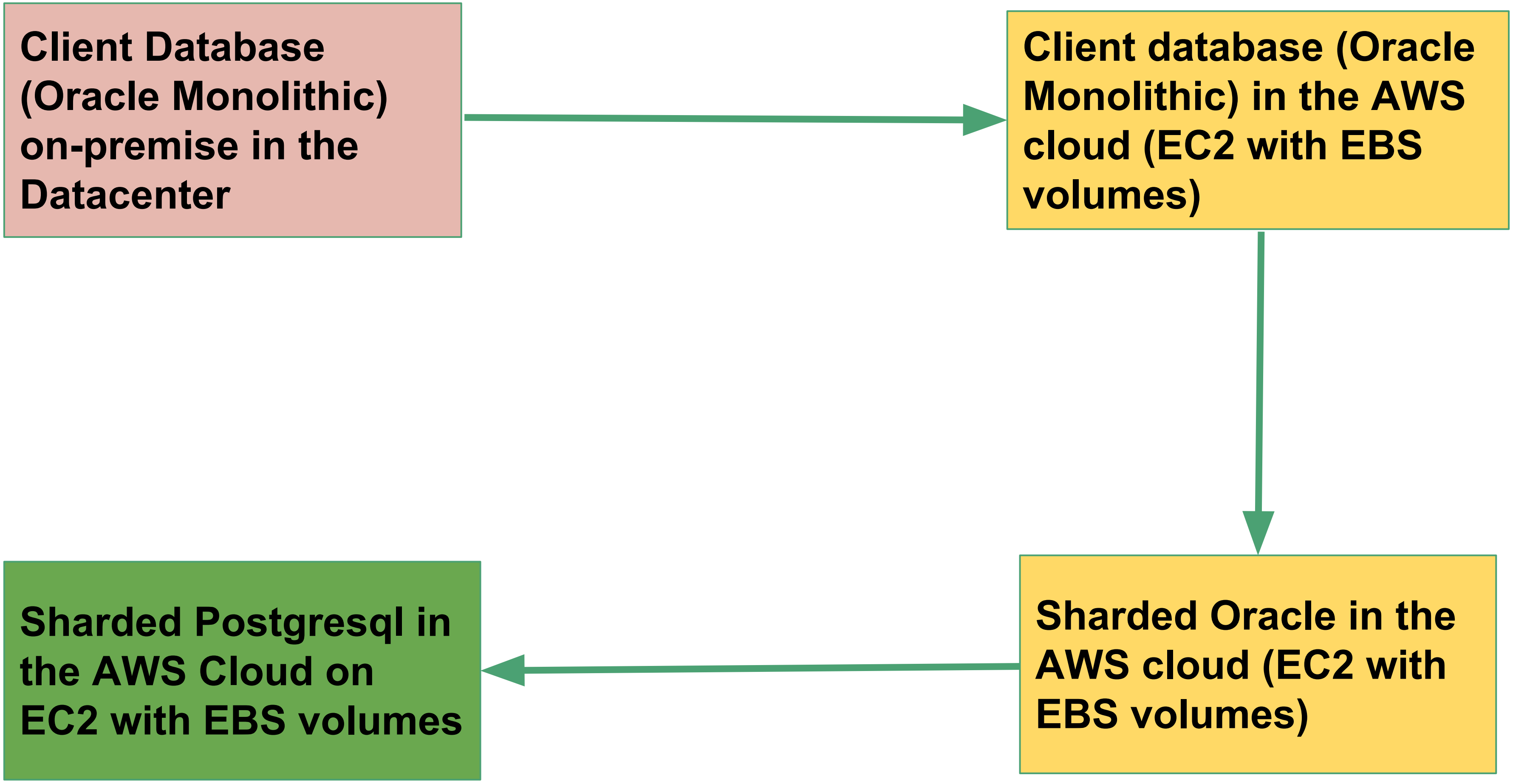
The proposed solution - Postgres

- Move to a sharded approach as there wasn't much commonality in the customers data
- Reduce load and optimise query performance
- Make good use of data distribution
- Reduce code complexity on front-end apps by maintaining query compatibility (as much as possible)
- Look at various variants of Postgresql such as Citus DB & PPAS (EnterpriseDB)

The proposed solution – AWS Cloud

- Needed to replicate the machine in the cloud
- No hardware available for that scale
- Snapshotting is an issue (~48hr for 1 snapshot)
- Need to do hardware/software benchmarks for combinations (DC m/c had 2.5TB)
- Change schema due to compatibility issues with PG
- Evaluate RDS (Postgresql)
- Evaluate T1 m/c in beta

Proposed migration strategy



Client Database (Oracle Monolithic) on-premise in the Datacenter

Client database (Oracle Monolithic) in the AWS cloud (EC2 with EBS volumes)

Sharded Oracle in the AWS cloud (EC2 with EBS volumes)

Sharded Postgresql in the AWS Cloud on EC2 with EBS volumes

Why Didn't RDS Work?

- Storage & file size limitation (6TB)
- No root access to database instance & underlying host
- DBA access with certain limitations
- Tuning DB is restricted to certain params
- Number of connections is limited by memory
- Cannot create separate tablespaces (Oracle)

Oracle features used

- Encryption
- Large file support for video files
- JSON and unstructured data support (Blobs)
- Partitioning and subpartitions (hash partitions)
- GIS support (for targeting in geo areas)
- Temp Tables (lots of them)
- Indexes (were large and not optimised)
- External Tables
- Redo logs (WAL logs)
- Merge / Upserts

Tools/Features used

- Orafce
- Ora2PG
- Schema conversion tool (SCT)
- Database Migration Service (DMS)
- pg_partman
- PostGIS
- JSON/JSONB Support
- FDW

Orafce

- Hosted at https://github.com/orafce/orafce
- BSD license
- Implements a bunch of useful functionality
  - Date functionality (trunc, round, days_between)
  - dbms_output – communicate with client
  - utl_file – filesystem functions
  - dbms_pipe / dbms_alert
  - PLVdate (business days)
  - PLVstr / PLVchr / PLVsubst
  - dbms_random
  - Varchar2/ Nvarchar2 support

Orafce - Pros / Cons

- Pros
    - Rich functionality
    - Easy-to-use out of the box
    - Actively developed
- Cons
    - Need to compile with the right libs
    - Some date ranges do not work (Oracle bug!)
    - Documentation sparse
    - Some functions may not be completely tested

Ora2Pg

- Hosted at https://github.com/darold/ora2pg
- GNU General Public License
- Migrates schema & data from Oracle to PostgreSQL
- Reads Oracle’s catalog and creates equivalent PostgreSQL objects
- Exports full schema in PostgreSQL-compatible format

Ora2Pg - Pros / Cons

- Pros
  - Migration templates & reports
  - On the fly migration
  - Supports SQL queries conversion
  - Supports PL/SQL to PL/PGSQL conversion
- Cons
  - Ignores hierarchical queries
  - Only syntax level conversions for functions, packages/procedures

Schema Conversion Tool (SCT)

- AWS tool
- Heterogeneous database migrations
- Can convert objects like functions, procedures, etc.
- Marks the non-convertible code
- Can convert application SQL
- Migration Assessment report

Ora2Pg vs SCT

- Both convert most of the objects
- Both support on the fly migration
- Ora2Pg - Spatial data types, partitions, DB links
- SCT - Functions, packages/procedures
- SCT - Extensive migration assessment report, information about schema conversion, manual changes required, references to PostgreSQL docs

Database Migration Service (DMS)

- To migrate data
- Homogenous & heterogeneous databases
- Supports CDC
- Can migrate schema objects required for data migration - tables, primary keys
- Tight integration with SCT
- One endpoint must be in AWS
- Be cautious when migrating LOBs, FLOATs, utf8mb4

PostGIS

- Support for geographic objects to the PostgreSQL object-relational database
- Distance between 2 points, area, perimeter functions
- Geometry types for Points, LineStrings, Polygons
- After transforming with ora2pg (worked mostly with some tweaks)

pg_partman

- Hosted at https://github.com/keithf4/pg_partman
- Postgresql license
- Useful in Archival / Query optimisations
- Time-based and serial-based table partition sets
- Sub-partitioning is also supported
- Child table & trigger function creation is all managed by the extension itself
- Optional retention policy

pg_partman

- Note about partitions, subpartitions and hash partitions in Oracle
- Partitions need to be pre-created in PG
- In PG, Partitions are just tables.
- Partitions can be separately indexed so queries can be made faster.
- Gotchas
  - Incompatible with PG 9.6 currently !!

Conclusion

- Postgresql has improved a lot in 9.x series
  - Replication support
  - Unstructured data / JSON support
  - FDW is useful for extensions
- Can support large tables and optimisations
- Indexing support is quite extensive
- Per query optimisation is still not possible
- Hash Partitions are supported
- Partitioning support needs to be improved
- Logical decoding would be useful for conditional replication

Q&A

Vinayak Hegde (@vinayakh / vinayakh@gmail.com)

Ravi Rayapati (@ravirayapati25 / rkrayapati@gmail.com)