Active-Active Replication

Presented to: Scale 17x

Dave Cramer
March 2019

CRUNCHY
Enterprise PostgreSQL

Dave Cramer

- Senior Data Architect @ Crunchy Data
- Major Contributor to PostgreSQL
- Working with PostgreSQL since 1999
- Maintainer of the PostgreSQL JDBC Driver
- Helping out with PL/R, some other small projects
- Real interest in where things like reactive Java and PostgreSQL are going

Crunchy Data

Leading provider of trusted open source PostgreSQL technology, support and training.

PostgreSQL
the world's most advanced open source database

Powering Innovation With The World's Most Advanced Open Source Database

CRUNCHY
Enterprise PostgreSQL

Active-Active replication

- What is it?
- Why do we need/want it?
- Symmetric-ds
- Different use cases
- What it doesn’t do
- The good and the bad ?

What is active-active ?

- Read/Write operations on more than one database instance
- Updates are propagated to all instances
- Single source of truth
- Sounds simple, right ?
- What happens if there are conflicts ?

Why do we want it

- Local performance
- Data close to the application

High Availability

- Two (or more) identically configured instances.
- Both accept writes
- Near zero downtime when failing over. No change to configuration of the new primary

Symmetric-ds

- Open source project by jumpmind.com
- https://www.symmetricds.org/
- https://github.com/JumpMind/symmetric-ds
- Replication toolkit

Symmetric-ds

- Written in Java
- Uses JDBC for database access
- Web app
- Asynchronous
- Unfortunately uses triggers and a log table
- Need to manage bloat in Postgres

Symmetric-ds

- Java and JDBC
- This allows the software to run on multiple platforms
- Enables multiple database vendors (relatively) easily

Symmetric-ds

- Asynchronous
- Synchronous replication over significant distances would slow the primary down
- Does introduce the possibility of data loss

Symmetric-ds

- Scales to thousands of databases
- Very configurable
  - Can configure which tables, columns or even rows
  - Can configured to push or pull data

Beyond Replication

- Zero(near) downtime upgrade of PostgreSQL
- Migration to PostgreSQL
- Cloud provider lock in

Upgrade on disk format

- Upgrading major versions of PostgreSQL can be painful
- Even pg_upgrade requires *some* downtime
- Change to heap checksums is not currently possible with pg_upgrade
- Admittedly this can be done with logical replication

Migration to PostgreSQL

- Migration from other database vendors to PostgreSQL
- Uses JDBC so works with a number of vendors
- Oracle, MSSQL, MariaDB, DB2 to mention a few

Cloud provider lock in

- Most do not allow binary replication
- Difficult to get ports other than 80 or 443 exposed
- Because Symmetric-ds is a web app it is relatively simple to replicate from one cloud to another

More than active-active

- Can be configured as single or bi-directional
- Can choose which tables to sync
- Has filters and transforms
- Can have one table(s) going up and another down

Central office with stores

- Each store has a local database for performance
- Each store can sell the item for a different price
  - For instance when they want to have a sale
- The resulting sale would be replicated to the head office in order to aggregate corporate sales
- When new items become available central office can push new items to the stores

Features

- How to deal with some of the details of replication
    - Conflict resolution
    - Transformations
    - Filters
    - Monitoring

Conflict Detection

- Various conflict resolution schemes
- Primary Key – will not allow a row to be inserted
- Changed Data – primary key plus only data changed
- Old data – All the existing row data will be used

Conflict Detection

- Timestamp – If the Primary key plus the timestamp column are equal
- Version column – If the Primary key plus the version are equal

Conflict Resolution

- Manual
  - Fix the data and mark it resolved
- Ignore
  - Ignore the batch or the row depending on config
- Newer wins
  - Either the source or destination with the latest change, based on either a timestamp or version column

Transformations

- Data moving from the source to the destination can be transformed as it moves
- This can be done when it is read from the source
- Or when written to the destination

Transformations

- The simplest and default is just to copy the data to the destination column
- Possible to change columns
  - Id -> account_id for instance
- Possible to remove NULL’s
  - NULL timestamp to now()

Transformations

- A column can be omitted
- Set a column to a constant value
- Set to a system variable
  - Date, timestamp, node_id (source or target), null,
- Previous value before updating
  - Auditing of changes ?

Transformations

- Additive transform. target = target + multiplier (source_new - source_old)
- Substring – expression is 0,5 which is passed to the java substring function
- Left – copies left most characters
- Bleft – copies left most bytes
- Many more

Transformations

- Bean Shell
- Executes the script in the expression
- Has access to column name, current value, old value, channel id, source node, target node and can execute SQL.

Filters

- Applied when loading the data
- Can be used to determine if the new data will be replicated
- For instance do not replicate new prices that are less than the old price

Monitoring

- Machine resources: CPU, Memory, disk space
- Replication status
  - Batch Error
  - Batch Unsent – data ready to be sent
  - Data unrouted – data ready to be batched
- Nodes offline

Monitoring

- Log file – Can be used with log aggregators
- Send emails when an error occurs

Panacea … not

- Does not replicate sequences
- This creates a problem if the sequence exists on both primaries
- So what happens: insert on one instance increments the sequence on that instance, subsequent insert on replica will fail with primary key failure (data will be inserted without using sequence

Panacea … not

- Solution to this is to increment sequences by 2 or more
- If you have 3 primaries you would have to have increment each by 3 and so on

Panacea … not

- Does not replicate DDL
- API allows for moving DDL over as well as data
- It is possible to lose data as this is asynchronous

In Summary The Good

- Java, runs anywhere
- Cross database
- Web app, easy egress and ingress
- Cloud migration, avoid lock in
- REST API

The Bad

- It uses triggers and they (symmetric-ds) don’t deal with bloat well. This is specific to PostgreSQL
- It’s,a bit complicated to setup
- Jumpmind has a paid (pro) version with a GUI
- The REST API is not complete enough to build a GUI
- GPL

How To

- https://info.crunchydata.com/blog/a-guide-to-building-an-active-active-postgresql-cluster

THANK YOU!

Dave Cramer
dave.cramer@crunchydata.ca