Experiences migrating a 7 TB tango archiver

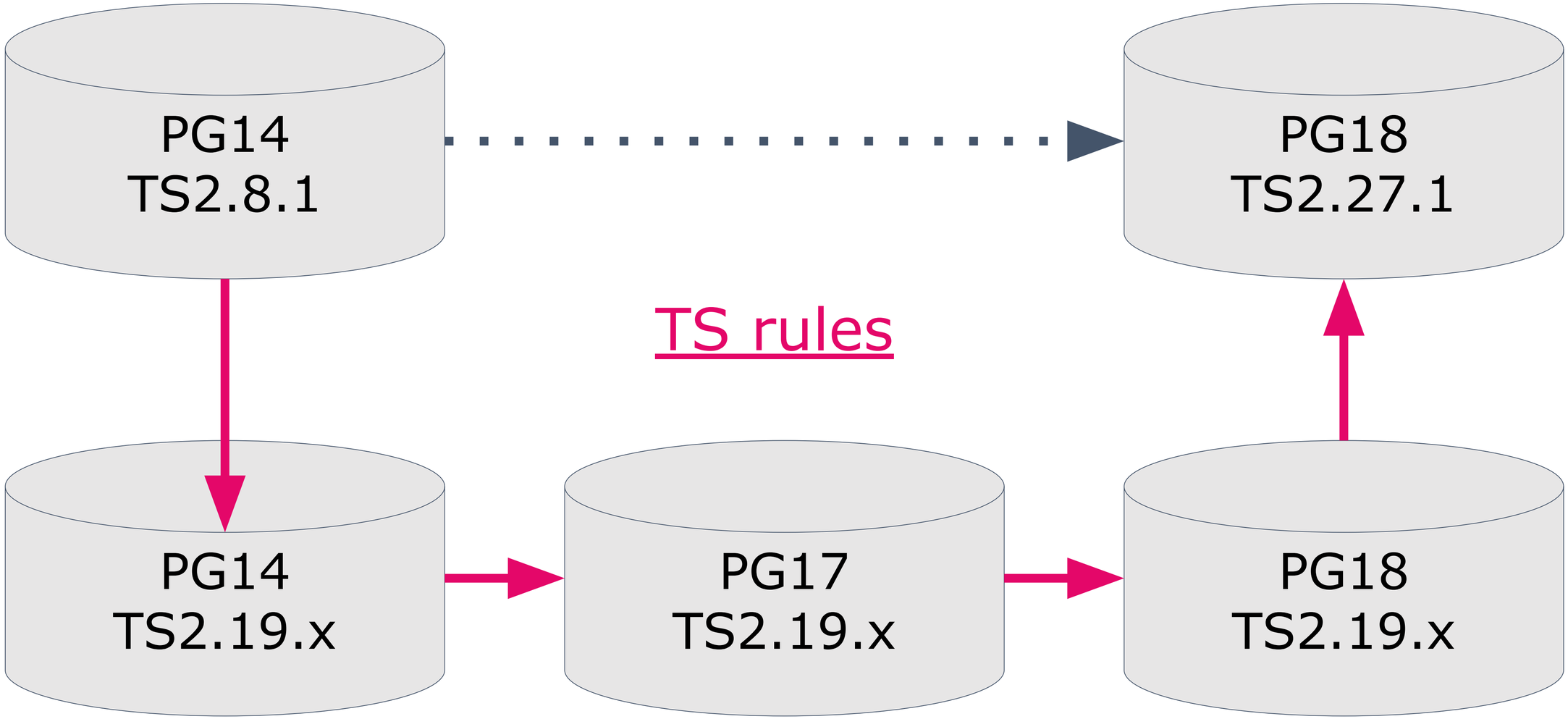
Tango Community meeting - HDBpp meeting - June 2026

Mauricio Zambrano

The SKAO EDA

- Engineering Data Archiver (EDA):
  - Hdbpp-timescale-project based
  - PostgreSQL (PG)
  - TimescaleDB extension (TS) for time series
  - Happily run the DB on Kubernetes with StackGres
- It keeps track of:
  - Devices properties
  - Important during commissioning
  - Users:
    - Engineers
    - Scientists

Migrating PG/TS between versions



Do you like risk?

- Investment:
  - High gain, 50% chance of loss
  - Low gain, 5% chance of loss
- Database:
  - Very long transaction that can fail?
  - Many short transactions that could be re-run?

Costs: money/time/risk of destroying a DB

Migration for a tango archive at LOW MCCS

- LOW:
  - MCCS
  - 10 stations
- 7TB compressed source database
- Installed from a helm chart:
  - Made by Timescale - delivered my team Himalaya
  - Support EOL - unsupported
  - Patroni (HA) enabled
    - Issues keeping replica up to date
    - Network/storage/wal sender/receiver issue

Migration path PG14->PG18

- Copy PV from source location to Pawsey
- Recover the database, using the same old helm chart
- PG14 on new location:
  - Some patroni hacking
  - Same problem at a newer location with more resources
- Use **something** to migrate data from PG14 to PG18
- Hypercore: new hybrid row-columnar storage engine in Timescale ~2025
- Data is compressed on source, size is really ~70TB

Something options - provided by PostgreSQL

- Pg_basebackup ~ 1month
- pg_dump: export/import ~ 1month
- psql … -c “COPY (SELECT * FROM mytable)” TO STDOUT WITH BINARY" | pv -trab | psql … -c "COPY mytable FROM STDIN WITH BINARY"
  - Good performance, compression, 1 thread
- Size is an issue, long, all or nothing

- Partitioning options:
  - Divide data by some time (hours/days)
    - On failure re-run the job, +1 threads

Something: partitioned version

- Partition last query on source, insert it on target, move to the next chunk - 1 thread
- TimescaleDB tool:
  - TimescaleDB parallel copy
  - psql … -c "COPY (SELECT * FROM mytable) TO STDOUT WITH **CSV HEADER**" | pv -B 32G | timescaledb-parallel-copy --connection … --table "mytable" --workers "**12**" --on-conflict-do-nothing --schema migration -auto-column-mapping --reporting-period 10s
  - Issues with image tables

Interesting findings

- You can export 70TB with almost with almost no RAM: use a unix pipe
  - PV tells you information on the pipeline
- SKAO average compression rates ~ x10

And more options…

Data pipeline platform

- Popular with data engineers
  - Ways to move data around - backfill jobs
  - Python based pipelines
  - Scheduler
  - Uses a PostgreSQL database
  - Opensource options:
    - Airflow, dagster, prefect

Dagster is kubernetes native, online free training. They also have a cloud product.

Dagster backfill pipeline

- Multi partitioning:
  - Timescale table name
  - Daily/hourly
- Dagster backfill pipeline:
  - Python files:
    - Assets (tables, data)
    - Resources (database connections)
    - Definition (putting all together, might include a schedule)
    - Sensors (dynamic functionality to modify behaviour)
- You can create dependencies between assets.

Running a pipeline

- “Materialize an asset” on the GUI/programmatically
- The dagster scheduler runs it
- Define concurrency: 32 jobs max
- Executed with a PostgreSQL account with 35 connection limit.
- Hourly partitions:
  - Good for big tables, bad for small ones
  - Opposite is real for daily partitions but very long txs
  - Middle ground: daily partitions but hourly partitions on the job

And sure, let’s make it harder, that was too easy

- We started a new PG18.1 database with TS 2.25.1
- New att_conf created from scratch
- Hydrate from the old EDA:
  - Lookup table for att_confs, att_errors
- Then copy everything again one more time:
  - Local migration database to the new production one

Lessons

- Data migrations are currently long and hard:
  - Size is an issue
  - Could we make them great again?
  - Risk of keeping old databases:
    - Security:
      - OS
      - Database
      - Extension
      - Migration
- Full flexibility with a data pipeline/orchestration platform
- att_conf does not preserve ids

pyHdbpp parquet backend

- Our LOW EDA has now 50TB
- We can’t growth forever:
  - Aggregation helps, image aggregation
  - Nobody is willing to just delete data
- Could we move data to another storage and still query it?
  - Yes: **parquet**, iceberg with an export job
  - Columnar oriented file formats with compression
  - Can be stored/queried from an S3 bucket:
    - TABLE_NAME/year=2026/month=02/day=01/hour=01/data.pqt
- How to query it:
  - Pyhdbpp, API
  - Duckdb, pyduckdb, pyarrow on your laptop