Redis vs PostgreSQL: Making the Right Choice

Chris Travers

February 2023

About Me

I have around twenty four years of experience with PostgreSQL and databases in many roles – developer, database hacker, dba, and more. I love PostgreSQL's versatility.

Thanks to

- TimescaleDB — Sponsoring my travel and work on this talk
- Adjust GmbH for where I got this experience
- Delivery Hero, for helping me solidify some of my thinking on this topic

Special Mention

OrioleDB, for collaboration on this topic regarding competitiveness of PostgreSQL and Redis on different workloads.

Agenda

Databases at Scale

Why this Talk

- There are many misconceptions about scalability
- True both of Redis and Postgres
- Both are good databases
- Both have critical scalability limits
- Guidance to create scalable systems is lacking

Database Use Changes at Scale

Most of us have been through at least part of this.
Using volume because it is easy. Same goes for velocity.

PostgreSQL at 1GB

- Black box
- No tradeoffs in theory vs practice
- Limited cases for indexes
- Theory-heavy, practice-light

PostgreSQL at 10GB

- Have to learn indexes
- Storage starts to matter
- Query efficiency starts to matter

PostgreSQL at 100GB

- Indexes really matter
- On-disk storage starts to matter
- Backups are no longer trivial
- Performance tuning starts here

PostgreSQL at 1TB

- Backups require specialized knowledge or frameworks
- Performance and vacuum tuning are critical
- Beginnings of physical limitations

PostgreSQL at 10TB

- Internals matter a lot
- A lot of performance management involves internals

Early on

- Databases are black boxes
- Performance is always good enough
- Costs are straight forward

Later

- Advanced features matter much more
- We have to reason from internals
- Bottlenecks due to implementation surface

Case Study

Why we moved a large Redis environment to Postgres at Adjust

Environment

- Many large Redis instances
- Queries total of 700k times per second
- Written to occasionally
- Sharded, replicated environment
- Fragile and an administrative headache

Reasons to Move

- Expensive hardware (RAM etc)
- Administrative nightmare
- Fragile and causes impact when things go down

Redis Internals and their Implications

What is Redis?

- Data structure manipulation layer
- Can be used for queues, key/value stores, and hashmap stores
- In-memory database
- Persistence is optional

Basic Redis Architecture

- Single threaded event loop
- Optimized for max performance of a single thread
- Some persistence tasks delegated to another thread/process
- LUA scripting runs as a separate client
- No parallelism

Redis Persistence and Replication

- Replication is similar to PostgreSQL streaming replication
- Persistence is optional
- Changes start in memory and then persist (usually to aof which is then separately rolled up)

Architecture Implications

- No parallelism
- Each new processor for queries needs full dataset in memory
- each write competes with reads on all replicas
- Extremely fast for one core, very hard to scale up
- Some workloads (queues in particular), don't work well with persistence

Redis Solutions

- Shard via Nutcracker
- HA vs Sentinel
- These add complexity very early on with Redis.

What is PostgreSQL

- A sophisticated, performant RDBMS
- Extremely extensible
- Scales up to large numbers of cores

PostgreSQL Architecture

- Multiprocess
- No multithreading but heavy use of IPC
- Built to scale up

Replication and Durability

- Transactions are persistent by default
- Replication tied to persistence
- Many options for replication (streaming, logical, etc)

Compared to Redis

- PostgreSQL is slower per core
- PostgreSQL scales up much more easily
- At scale, PostgreSQL is much easier to manage

At massive scale

- There are few great tools for distributed data on PostgreSQL
- Often folks have to write their own
- This happens much later than with Redis
- Costs are usually less.

Scenario

- Large sentinel plus nutcracker setup
- Legacy of earlier time when redis was used more
- Redis was seen as a reliability concern due to complexity
- PostgreSQL could do what we needed it to do, cheaper.

Redis as Cache

- Postgres and Redis sit side by side
- Postgres is authoritative
- Recently used data cached on Redis
- Problems include cache invalidation
- Approaches include application-level caching and logical replication

PostgreSQL as Key/Value Store

- PostgreSQL replaces Redis
- Semistructured data goes into a key/value table
- Many options including JSONB
- Harder to replace Redis as a queue

Conclusions

- Redis and PostgreSQL are different, and have different limitations
- Redis is faster under small read-mostly workloads, and PostgreSQL scales better
- PostgreSQL can replace Redis in many workloads
- Cost depends on many other factors

Recommending Redis for:

- Redis shines best for small, read-mostly workloads
- Larger data-sets, and those with heavier writes require a lot more complexity.
- Cases where timing out entries is helpful.
- Great example is distributing authentication token information for large web applications.

Recommending PostgreSQL for:

- Large datasets
- High velocity datasets
- Wherever other PostgreSQL data should be able to be joined against the data.

Questions

Questions?