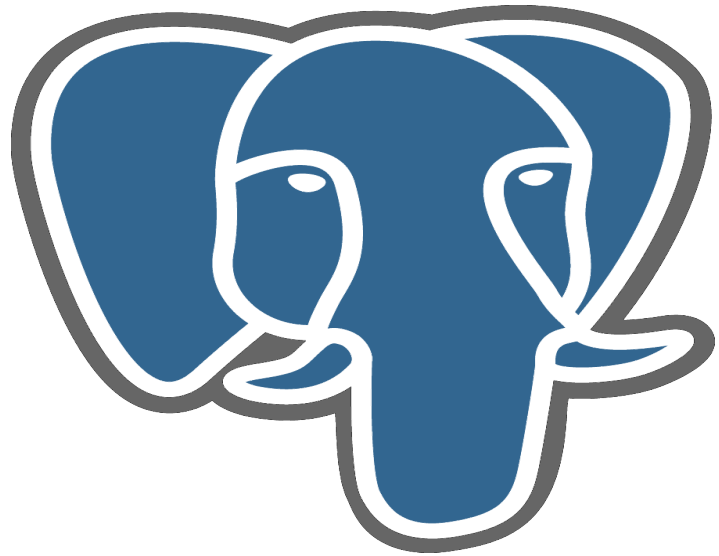
PostSQL
Using PostgreSQL as a better NoSQL

NoSQL matters 2013, Cologne

—-

Hannu Krosing
hannu@2ndQuadrant.com



Hannu Krosing

- Using Linux, Python and Postgres(ql) for 20+ years
- First (and for some time the only) DBA/Database Architect at Skype
- from 2007 full-time PostgreSQL consultant specialising in Scalability, High Availability and Performance
- Technical Consultant with `http://www.asi.ee`, the VC investing company of the Estonian founding engineers of Skype

2ndQuadrant

- a PostgreSQL consulting and development company with international reach with branches in UK, Germany, Italy, France, USA, Nordic Countries, South America, Australia, . . .
- many Major Contributors and Contributors to PostgreSQL development and development of related tools.
- offering ”follow the sun” 24/7 support with people in enough timezones to always have somebody awake

more at `http://www.2ndquadrant.com`

PostgreSQL

- Object-Relational database built for extensibility
- Up to 1995 Postgres used a language called PostQUEL

```
retrieve (DEPART.name)
  where DEPART.pietro
        NOT-IN {
          D.pietro
          from D in DEPART
          where D.name != DEPART.name
        }
```

So what is SQL

Even ANSI/ISO ”standard” SQL is a complex beast, consisting at least of

- DDL - Data Definition Language (CREATE)
- DML - Data Management Language (INSERT UPDATE DELETE)
- DQL - Data Query Language (SELECT)
- ACL - Access Control Language (GRANT, REVOKE)

SQL

There are also ”standard” ways to do

- XML processing
- Foreign Data Wrappers (XML-MED)
- embedded procedural language

SQL

Problems:

- If you are bound by a ”Standard” it is harder to innovate.
- And to make it worse, once using ”An SQL Database”, many companies aim at being ”database independent”

SQL

Problems:

- If you are bound by a ”Standard” it is harder to innovate.
- And to make it worse, once using ”An SQL Database”, many companies aim at being ”database independent” **as if it were a good thing**

NoSQL

what is ”No”SQL

- CRUD (create-read-update-delete)
- usually at least DML and DQL, with some of DDL and ACL, different syntax
- no need to be ”SQL compatible” , so you can optimise for your use case
- mosttimes ”just storage”, with growing number and complexity of near-data OLTP operations (update-if, . . .)
- for OLAP, usually some form of map-reduce

NoSQL ease of use

- generally easy to get started,
- fast and flexible development - no data model syncing (NoDBA)
- fun as you get to play with what would be hidden in the internals in a classical database

NoSQL ease of use

- generally easy to get started,
- fast and flexible development - no data model syncing (NoDBA)
- fun as you get to play with what would be hidden in the internals in a classical database and **reimplement** large parts of it

NoSQL scalability

- as data module is ”just storage” it is easy to scale
- we did something similar with more complex schema at Skype using pl/proxy

PostgreSQL

A robust and powerful data processing platform which is - among other things - an SQL database

If you are using it as **only** an ”SQL database”, or even as ”storage” you are probably doing it wrong.

- safe BSD licence controlled by no single entity
- no need to worry about somebody ”buying up” the company

Why PostSQL

Achieving the good parts of NoSQL without throwing away good parts of PostgreSQL
I use *Post*SQL instead of *No*SQL
(A ”No” is a ”No” , not ”Not Only”)

PostgreSQL standard SQL features

- PostgreSQL is complex extendable data storage and processin platform with
- robust ACID
- good OLTP performance, with simultaneous good OLAP performance
- writers dont block readers
- readers dont block writers
- writers block each other only when writing the same row

PostgreSQL nonstandard extensions to SQL

- user defined data types
- user defined functions
- user-defined indexes, generic extendable indexes
- in pluggable languages modues (C, SQL, plSQL, python, perl, R, java, v8, . . .)
- adjustable durability levels - async, sync, syncrep commit

PostgreSQL as a ”Document Database”

”Structured” or ”Documen” data types

- XML
- JSON
- hstore

Write/Integrate your own

- like wrap BSON

PostgreSQL as ”Document Database”

PostgreSQL-s generic support for functional and conditional indexes lets you index ”into” the structured data types

You can also use inverted indexes (GIN, GiST) to have multiple index pointers pointing to same objects

And you can expose your ”Documents” as relations via set-returning functions,

indexing Documents/Attributes

- there is functionality built-in for structured types
  - XML/Xpath
  - JSON attribute extraction
  - hstore key/value indexing

Integrating external data

- set returning functions
- Foreign Data Wrappers (FDW)
- easy to prototype using `http://multicorn.org/`

Scaling PostgreSQL @Skype

- We started at SKype with single PostgreSQL database
- This worked fine for used authent storage
- Added another for accounting and paid calls to POTS
- Did not work that well once we started having serious traffic.

Scaling PostgreSQL @Skype

- Fortunately we had decided to have a ”NoSQL” client API.
- Client always communicated with database using a call in for

```
SELECT <somefunction>(<arguments>);
```

call a funtion get a (set of) data back

Scaling PostgreSQL @Skype

```
SELECT <somefunction>(<arguments>);
```

- This allowed me to reorganize large portions of the database without clients noticing anything and with no need to schedule downtime for upgrades
- As we later found out In Java word this is called Service Oriented Architecture (SOA)

Scaling PostgreSQL @Skype

- So we started adding database servers.
- We did the usual dance of horisontal splitting of data by function onto onto several servers
- this allowed us to still have internal consistency within same server
- and for the few places we needed cross-database data, we did either
  - remote calls using pl/python for single value CRUD
  - replication of some tables needed to be queried everywhere

Scaling PostgreSQL @Skype

- Around 2004 we saw that we were running into a ceiling on what we could do with our current way of scaling.
- Roughly the same time Google started with BigTable and a few years before Amazon SimpleDB.
- But as we had only a handful of database people at this time, we could not afford to start designing a database from ground up.

Scaling PostgreSQL @Skype

- And the standard Master-Slave read scaling did not work, as we had a write-mostly database.
- We had to scale writes, meaning **we needed sharding**

Scaling PostgreSQL @Skype

But as we were using excluseively function API from clients, we could do the sharding in these functions
In the beginning we used pl/python functions for sharding

```
CREATE FUNCTION get_email(user)
LANGUAGE plpythonu AS $$
import psycopg
con=connect(getconnection(hash(user))).
cur.execute("select get_email(%s)", use
return cur.fetchall()[0]
$$;
```

pl/proxy

As we were doing it more and more, I suddenly had an ”Aha!” experience and a special sharding language was born

```
CREATE FUNCTION get_email(user)
LANGUAGE plproxy AS $$
CLUSTER users
RUN ON hashtext(users)
$$;
```

Extensions by Skype

- in-database queues, replication, online ETL - skytools package
- partitioning language pl/proxy
- connection pooler pgbouncer

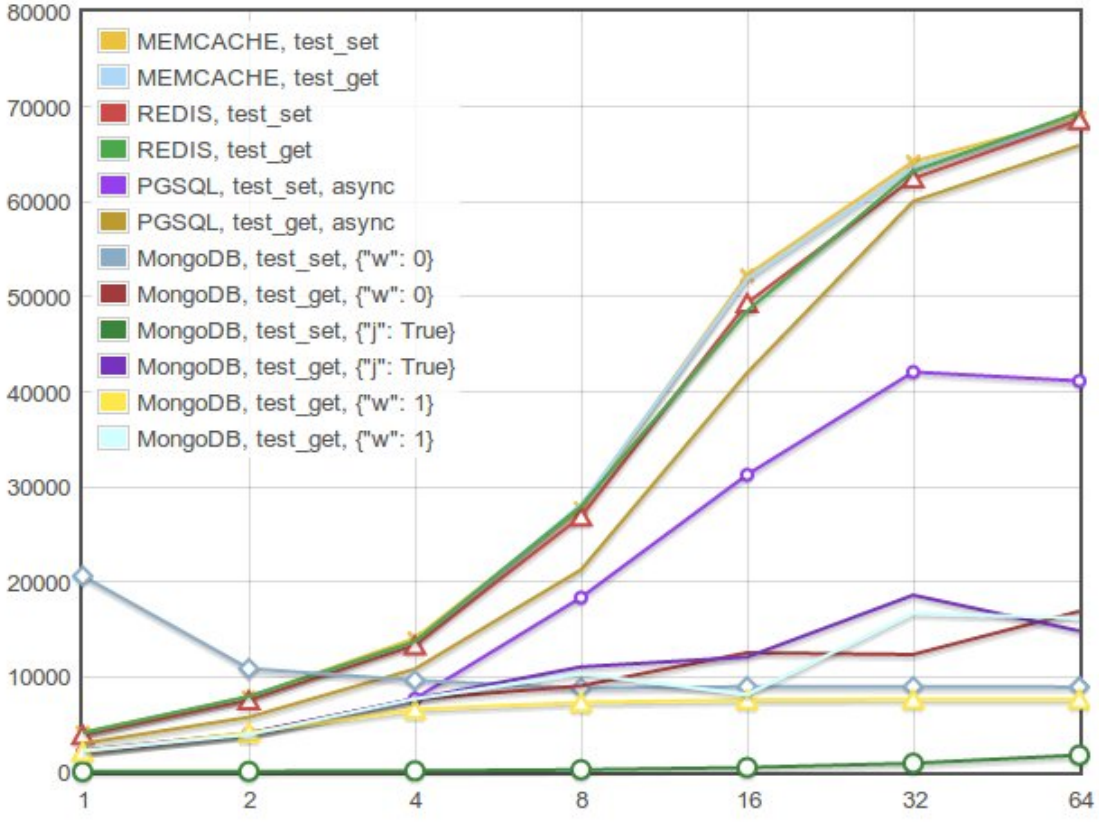
performance

The followiong is a result of a small study I did comparing

- memcache
- redis
- PostgreSQL
- MongoDB

On an relatively low level server. Single Intel CPU 4, cores with HT
Dataset is a few Gigabytes so it fits all nicely in memory

performance



1 Server with 1 to 64 clients, Client(s) and server on separate host
minimum data size: 1188, max size: 2601, average size: 1874

Vertical axis is ops/transactions per second
Horisontal is number of parallel clients

PostgreSQL Durability modes

- async
- synchronous
- synchronously replicated

Conclusion

For many use cases you can actually hack no-sql type behaviours into PostgreSQL using its extensibility features.
Often the overhead of managing data in multiple databases is more than the advantages of the other store being faster.
You can do ”NoSQL” inside and around a hackable database like PostgreSQL, not just as a separate one.

Questions?

Hannu Krosing, hannu@2ndquadrant.com

- read more on PostgreSQL Administration at http://www.2ndquadrant.com/books/
- and there is an Upcoming book on PostgreSQL server Programming also coming from Packt