Logical replication with pglogical

Moving the same old data around in new and exciting ways

© 2ndQuadrant 2016

pglogical

It's row-oriented replication

Really, that's pretty much it. The rest is details.

Done now?

© 2ndQuadrant 2016

pglogical

Open source – PostgreSQL license

Submitted to 9.6

Generic, re-usable, no custom PostgreSQL

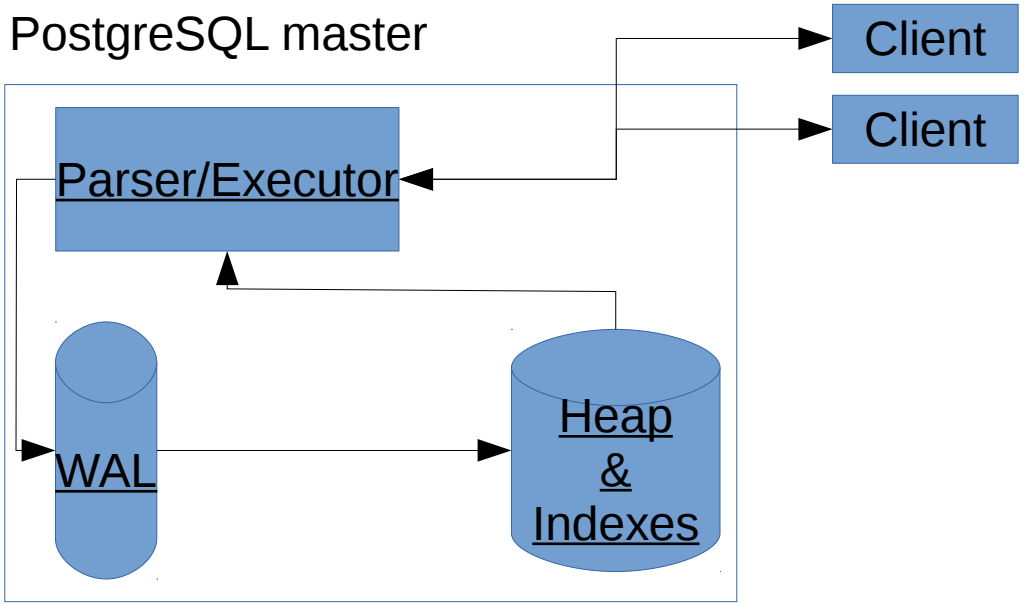
© 2ndQuadrant 2016

Architectures

- Standalone (no replication)
- Physical replication (block level)
- Logical replication (row level)

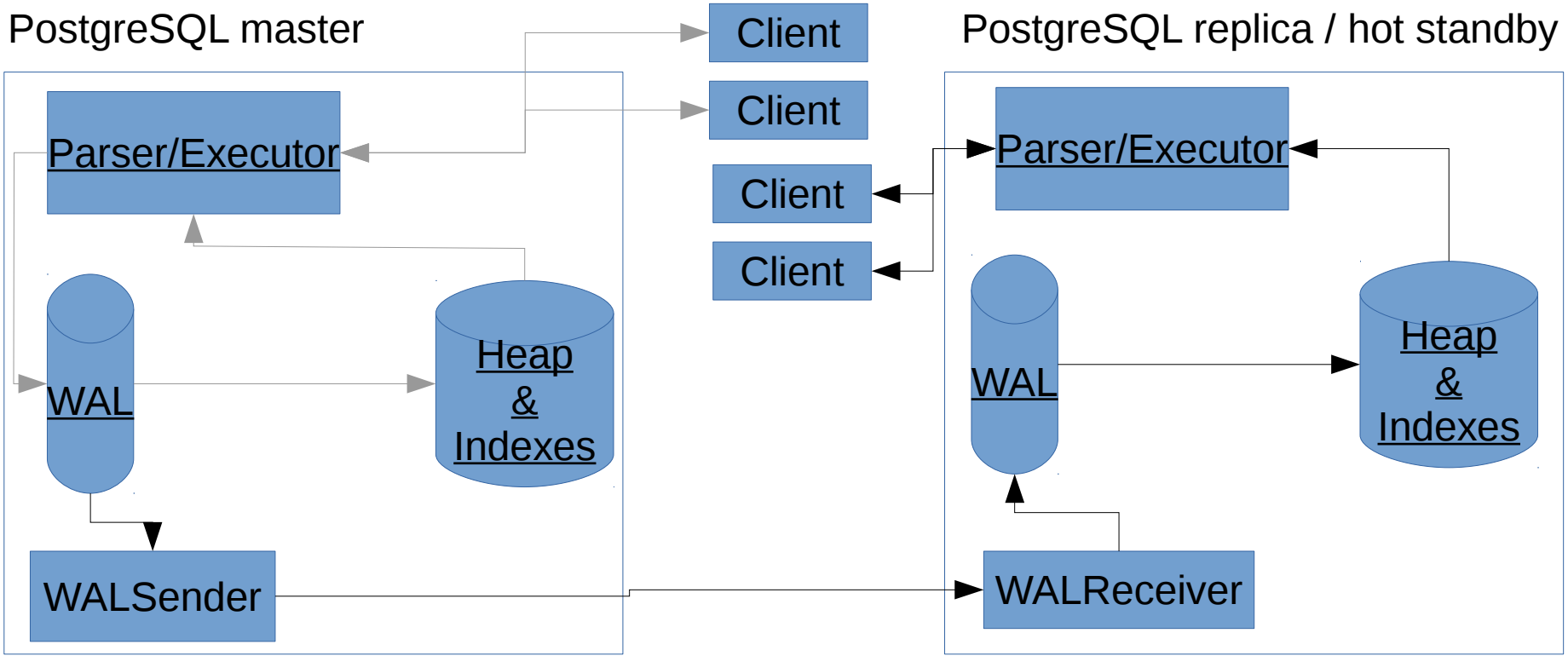
© 2ndQuadrant 2016

Standalone PostgreSQL



© 2ndQuadrant 2016

Physical replica & hot standby



© 2ndQuadrant 2016

Confused yet?

© 2ndQuadrant 2016

“Physical” replication

Copies *everything*:

- Every database
- VACUUM
- Index updates
- ….

© 2ndQuadrant 2016

“Physical” replication

Fast to apply changes to replicas.

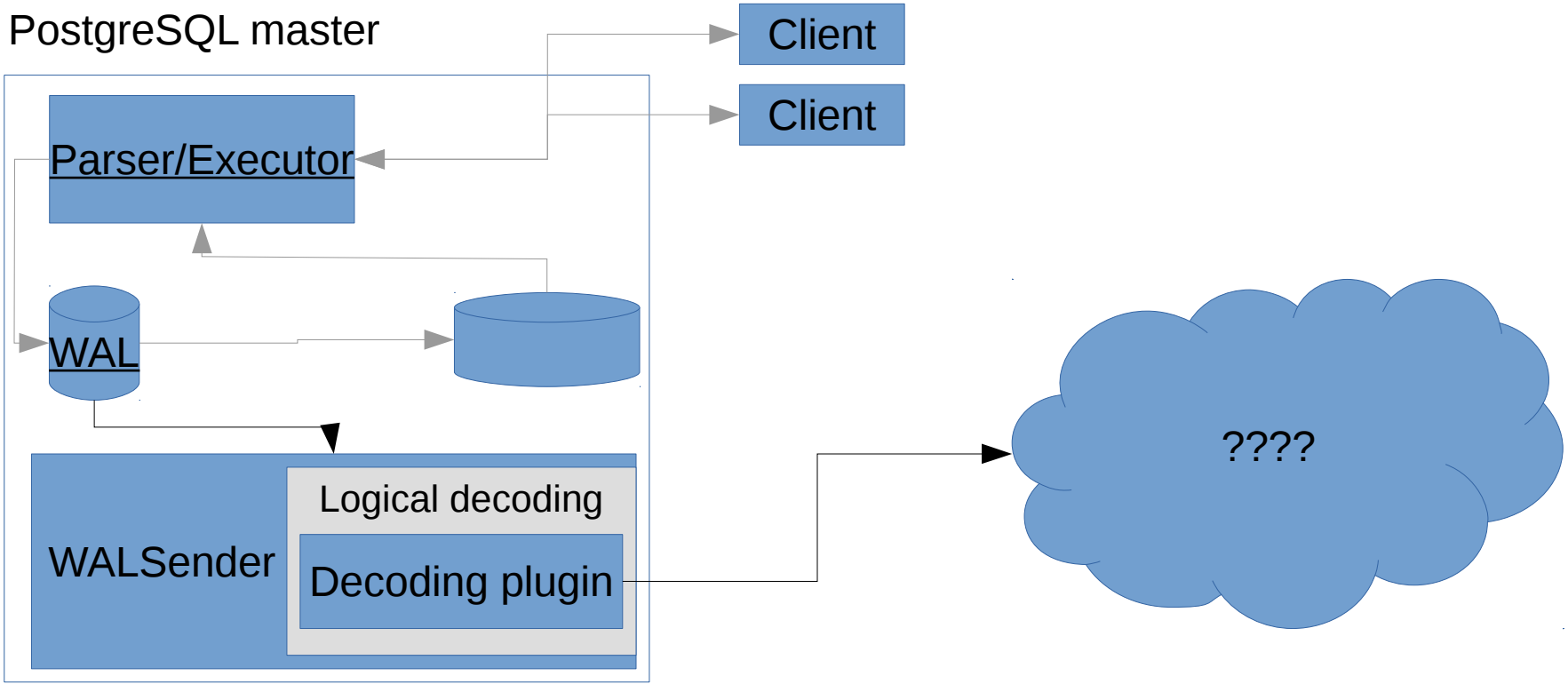
Bandwidth-hungry.

All-or-nothing.

Hot standby limitations

© 2ndQuadrant 2016

Logical decoding



© 2ndQuadrant 2016

Logical decoding

Collects just *row values*

No VACUUM traffic, index updates, etc

Can generate text-format values for replication to other Pg versions etc

© 2ndQuadrant 2016

Logical decoding

- Useful for logical replication
- … but not just replication
  - Intrusion detection
  - Search
  - Messages buses
  - ….

© 2ndQuadrant 2016

Many logical decoding plugins

- **pglogical_output**
- **BDR** output plugin
- The demo **test_decoding** to stream SQL
- **decoder_raw** and **receiver_raw** to stream and apply SQL
- github.com/ildus/**decoder_json** and github.com/leptonix/decoding-json to stream JSON
- github.com/xstevens/**decoderbufs** to stream as protocol buffers
- github.com/confluentinc/**bottledwater-pg** to stream to Kafka
- ….?

© 2ndQuadrant 2016

pglogical_output

Make it *easy* and *generic*:

Both json & fast native proto

Selective replication, metadata, etc

Use it without writing a bunch of C

© 2ndQuadrant 2016

Logical *replication*

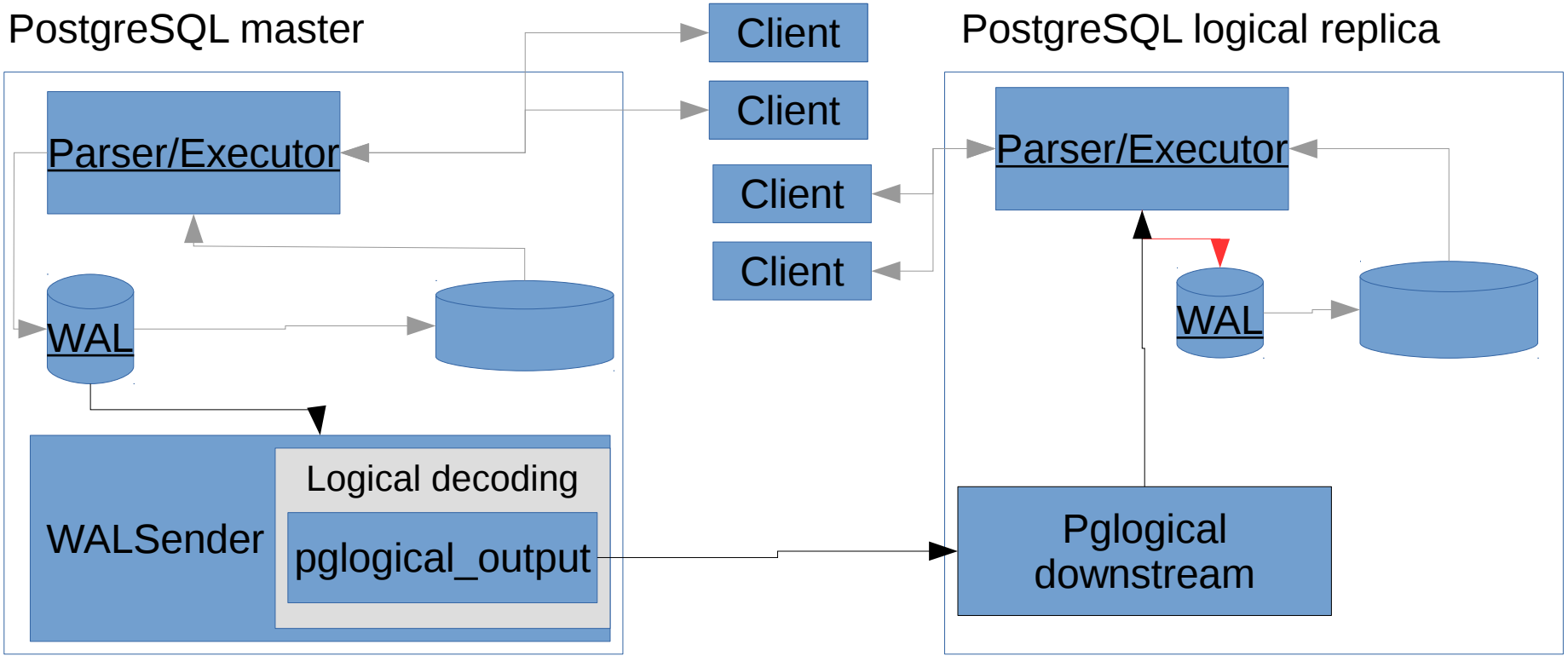
Selective – just the DBs/tables you want

More flexible standby with read/write tables, no query cancels

Not just 1:1 – sharding, data gather, ...

© 2ndQuadrant 2016

Pglogical



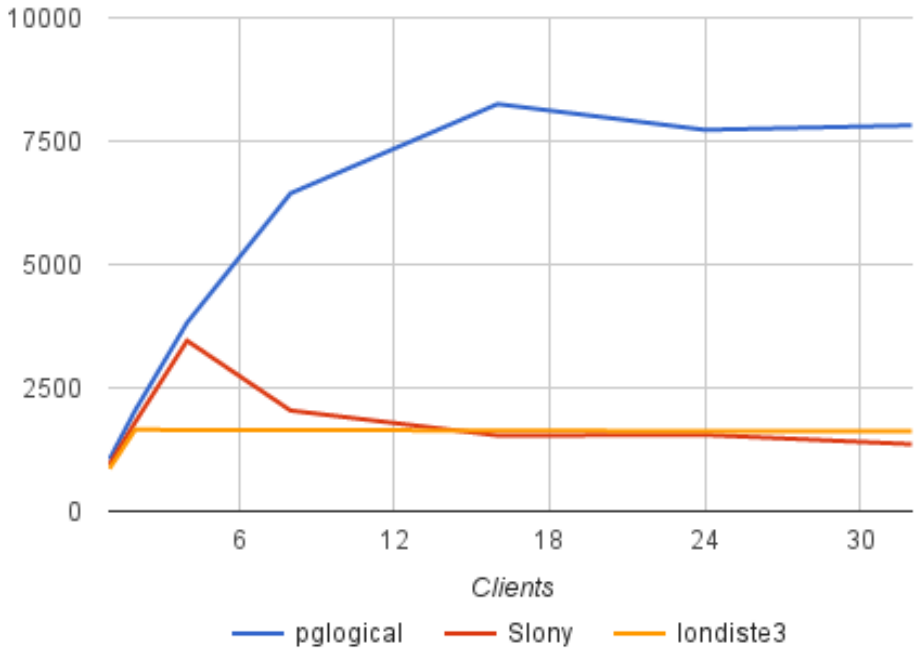
© 2ndQuadrant 2016

Pglogical: now

Selective replication
Crash safe
Downstream replica writeable
No query cancels on downstream
Efficient COPY-like apply process
No-downtime cross-version upgrade

© 2ndQuadrant 2016

Pglogical: performance now



© 2ndQuadrant 2016

Pglogical: future

Filter rows by WHERE clause
Logical and Physical Failover
Continuous ETL, transform
Continuous Data Warehouse ingestion
Automatic DDL replication
….

© 2ndQuadrant 2016

Sending data to apps

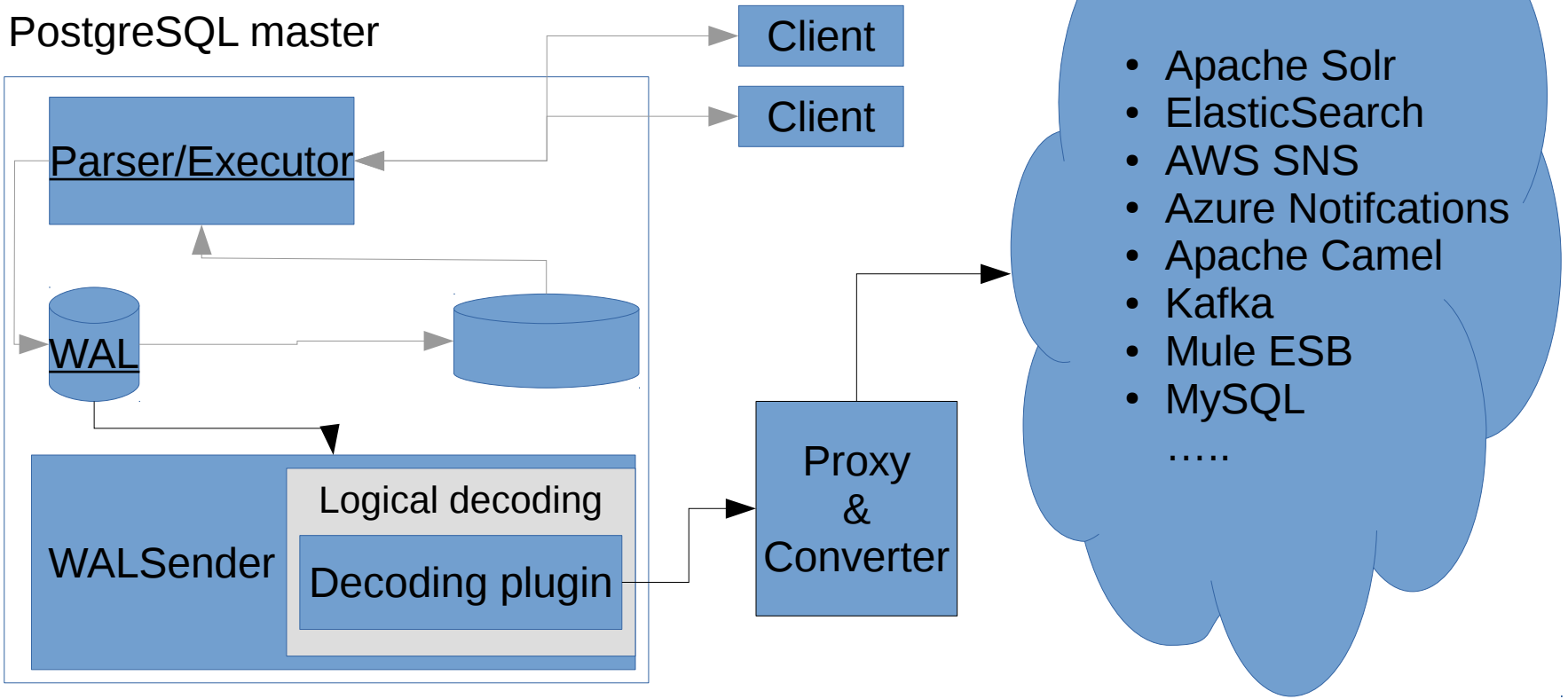
Take json or native output from pglogical_output

Proxy it to the app with a script

Ingest it into the app

© 2ndQuadrant 2016

Decoding to apps



© 2ndQuadrant 2016

Demo: solr

Take json or native output

Proxy it

Ingest it

© 2ndQuadrant 2016

Demo: solr

The code is simple but not brief enough to list here.

https://gist.github.com/ringerc/f74a12e430866ccd9227

© 2ndQuadrant 2016

Demo: solr

The process:

- Make a normal psycopg2 connection
- Create a replication with `pg_create_logical_replication_slot` slot if it doesn't exist
- Loop over `pg_logical_slot_get_changes` to fetch the change stream
- Accumulate a whole transaction's worth of rows
- Transform the JSON from each call into something Solr will understand
- Send the transaction to Solr over http

© 2ndQuadrant 2016

Initial database state

No good way to send rows already in the database when we set up decoding.

Workaround:

```
COPY (SELECT row_to_json(x)
FROM my_table x) TO stdout
```

© 2ndQuadrant 2016

Initial state

Ugly? Very. Plenty of room for improvement.

© 2ndQuadrant 2016

pglogical pitfalls

- Unused slots can fill pg_xlog
- DDL isn't replicated yet
- Serial streaming of big xacts causes latency
- Big xacts need extra disk space
- Sequences not replicated yet

© 2ndQuadrant 2016

Using pglogical

I won't repeat the docs.

Yes, there are docs.

The pglogical_output protocol is documented too, for app devs.

© 2ndQuadrant 2016

Slots: a public service announcement

Replication slots *prevent the server from removing still-needed WAL from* `pg_xlog`.

An abandoned, unused slot can cause `pg_xlog` to fill up and your server to stop.

Unused logical slots also create bloat in the catalogs.

© 2ndQuadrant 2016

Slots: a public service announcement

Add `pg_replication_slots` replay lag to your monitoring and alerting system. Use `pg_xlog_location_diff(...)`

You'll already have alerts on `pg_xlog` disk space, of course. Right?

Just like you regularly test your backups.

© 2ndQuadrant 2016

Questions?

© 2ndQuadrant 2016

Questions?

Q: Can pglogical be used to receive data from non-PostgreSQL sources and stream it into PostgreSQL?
A: Not *yet* but there's room to support it in the design of the receiver. Good idea.

Q: Can I replicate to non-PostgreSQL databases?
A: No, and the pglogical downstream isn't designed to do that. You could use the pglogical_output plugin to provide the data extraction and streaming facilities you need to send data to your own downstream though.

© 2ndQuadrant 2016