Why your PostgreSQL 9.0 cluster needs pgpool-II

It works, but there are things you must know

Jean-Paul Argudo
Dalibo
PostgreSQL Expertise Company
Paris, France

PgDay.Eu 6-8 december 2010
Thanks...
Tatsuo & SRA OSS guys

Special thanks to Tatsuo Ishii, lead developer of pgpool tools.
  ■ Wrote lots of wonderful tools for PostgreSQL
  ■ Reviewed those slides
  ■ Gave me some benchmark results I couldn’t prepare on time

Thanks also to all others participating in this effort:
Nozomi Anzai, Devrim Gunduz, Guillaume Lelarge, Haruka Takatsuka, Akio Ishida, Toshihiro Kitagawa, Yutaka Tanida, Tomoaki Sato and Yoshiharu Mori
Most of them are from SRA OSS Incorporation, Japan.
Jean-Paul Argudo, owner of Dalibo

Founder of PostgreSQL.fr website (February 2004)

Co-founder of PostgreSQL.fr non-profit (since February 2005)

Board member (treasurer) of PostgreSQL Europe (since 2008)
About...

...Dalibo

- PostgreSQL expertise in France since 2005
- More than 200 customers, from dotcoms to real big companies, including public institutions
- Participation in the CNAF migration to PostgreSQL (OMG! you’re in the wrong talk ?!)
- Consulting missions, trainings and support contracts
- Some logos of customers you may know?
Thanks to all speakers, staff and sponsors

Ground helpers are *AWESOME*, special thanks!
Presentation of pgpool-II
What it does

- At first, developed for connection pooling
- Replication
- Master/Slave mode
- Load balancing
- Automatic failover on desync detection
- Online recovery
- Parallel Query

pgpool-II does *lots* of things
Acts as a pooler: whenever we receive M connections from clients, only N connections will be used.

Usefull to limit incoming connections **without** sending errors to the clients: exceeding connections will be queued.
Presentation of pgpool-II

Connection Pooling
Based on statements replication: writes queries are sent to all backends, read queries can be balanced

Caveat: don’t let users access to backends directly!
Presentation of pgpool-II
Replication and Load-balancing of reads
Presentation of pgpool-II
Master/Slave Mode and Load-balancing of reads

- Works with any master/slave replication system (Slony-I, Londiste, Streaming replication, Bucardo, whichever you like) (despite I didn’t test all)
- Write queries are sent to the master only, aka `backend_*0`
- A new sub-mode in version 3 (september 2010) to handle Streaming Replication specifics
Presentation of pgpool-II
Master/Slave Mode and Load-balancing of reads
A mechanism exists to verify the sync between backends.

When a node is detected as down, a script is executed and arguments passed to it (failover.sh).

Depending the mode pgpool-II runs in, the actions to perform in the script are different. We’ll focus on this later.
Presentation of pgpool-II

Failover

1. Failover begins when the Master fails.
2. PGP initiates a failover.
3. Replication system is interrupted.
4. Writes & reads continue with Slave.

APP

PGP

Master

Replication system

Slave

Master
Online recovery can be done without restarting pgpool.

When a PostgreSQL backend needs to be restored, we can execute one pgpool-II command to restore it (`pcp_recovery_node`).

It uses pgpool local script (like `basebackup.sh`) to:
- Perform an hot backup of the master and send it to the node to be restored.
- Create remotely the `recovery.conf` script in the `$PGDATA` of the restored node.

And then starts the remote restored node thanks to the `pcp_remote_start` script.

Caveats: `basebackup.sh` and `pcp_remote_start` have to be located in the `$PGDATA`. 
Presentation of pgpool-II

Parallel Query

- Allows to virtually cut your data among multiples servers

- Needs a system database, called \texttt{systemdb}\ for pgpool-II to store information about your data

- Partitioning your data is based on simple functions of pgpool-II to know where to ask for

\begin{verbatim}
CREATE OR REPLACE FUNCTION pgpool_catalog.dist_def_accounts (val ANYELEMENT)
RETURNS INTEGER AS 'SELECT CASE WHEN $1 >= 1 and $1 <= 30000 THEN 0
WHEN $1 > 30000 and $1 <= 60000 THEN 1
ELSE 2
END;
\end{verbatim}

- To be used with quite large data sets
We’ll focus on those features on pgpool-II

- Master/Slave mode
- Load balancing
- Failover
- Online recovery
Clustering PostgreSQL with Slony-I

Schema

Master

\[\text{Slony-I} \rightarrow \text{Slon daemon} \rightarrow \text{Slon daemon}\]

Slave
Clustering PostgreSQL with Slony-I

Pros/Cons

Pros

- Write performance is good (10 to 20% overhead)
- Automatic failover of slaves
- Connection pooling and load-balancing: boost read performance

Cons

- Asynchronous replication, with a large delay
- No DDL replication, not easy to propagate DDL changes (slonik)
- No large objects replication
- Not that easy to configure and maintain
Clustering PostgreSQL with Slony-I
Replication delay with Slony-I

Courtesy of Tatsuo Ishii, SRA OSS Japan
At least for four reasons

- Slony-I can interact between PostgreSQL versions (same does Londiste and Burcardo). SR cannot. Even architectures have to be identical (32/64 bits)
- Slony-I can replicate parts of database. SR cannot, it replicates the whole PostgreSQL "server" (aka "cluster" aka "$PGDATA")
- Slony-I has cascading replication. SR doesn’t (yet (I hope))
- Because Slony-I is cool!

I think Slony-I will be used more specifically in the future, for what it does
Clustering PostgreSQL with Streaming Replication

PostgreSQL 9 with Streaming Replication

Master

Streaming Replication

Slave

WAL sender

WAL receiver
Clustering PostgreSQL with Streaming Replication
Low write overhead

Courtesy of Tatsuo Ishii, SRA OSS Japan
Clustering PostgreSQL with Streaming Replication
Low replication delay

Courtesy of Tatsuo Ishii, SRA OSS Japan
Clustering PostgreSQL with Streaming Replication

Pros

- Low write overhead
- Low replication delay
- Transparent replication: even DDL and large objects are replicated
- Easy to manage
Clustering PostgreSQL with Streaming Replication

Cons

- No automatic failover
- No connection pooling, no balancing
- The standby dislikes all write attempts and some other SQL commands, you’ll need to filter things sent to the standby
Pros

- Write performance is good (10 to 20% overhead)
- Automatic failover of slaves
- Connection pooling and load-balancing: boost read performance
- DDL and large-objects replication
New master/slave "sub_mode" dedicated for Streaming Replication and Hot Standby

Intelligent load-balancing": if the slave is far "more than delay_threshold bytes, then read queries are sent to the master only

Adding standby servers without stopping pgpool-II (pcp_attach_node new method)

Scales very well adding slaves!
Problems and solutions

Documentation

**Problem**
- pgpool-II documentation is sometimes hard to understand
- the documentation exists only in japaneese and english

**Solution**
- A patch for the english version of the documentation comes from Dalibo
- I’m translating it to french (status 70%)
- Maybe I’ll translate it to spanish too
Problems and solutions

Industrialization

**Problem**

- Examples on the net are often too simple
- Leads to headaches when the architecture to implement becomes complex

**Solution**

- Customer’s projects feedback from Dalibo to come: scripts and HOWTO’s
- Some patches (sent & currently in progress) to industrialize pgpool-II easier
- We shall study how to use `repmgr` maybe in our scripts
Problems and solutions
How to not balance

**Problem**
- You may need to *NOT* balance queries

**Solution**
- Add a specific tag anywhere in the query not to be balanced: /*NO REPLICATE*/ (poor)
- and/or use explicit transactions like
  
  ```sql
  BEGIN […]my SQL transaction[…] COMMIT;
  ```
  (better!)
Problems and solutions
JAVA app servers & autocommit on

**Problem**

- Most JAVA app servers are in autocommit to on
- pgpool-II won’t then balance anything, since we’re on implicit transactions

**Solution**

- First, set autocommit to off
- Second, learn to code transactions in SQL
- Third, don’t be transactional if your SELECT can be read anywhere
**Problem**

- Any function could create writes. Like `SELECT create_user('jpargudo');`
- Those functions will be balanced by default because of the `SELECT`... Even on the standby server(s)

**Solution**

- Declare any write function into `black_function_list`
- OR declare functions that can be balanced in `white_function_list`
- **NEVER** use both at the same time
- If you want any function NOT to be balanced put any unexisting function name into `white_function_list` and nothing in `black_function_list`
Problems and solutions

Automatic failover

**Problem**
- If pgpool-II detects anything wrong on a node, it executes `failover.sh` (or any script you configured)
- A non-reachable node is not a down node sometimes

**Solution**
- AVOID automatic failover. Prefer sending messages in your `failover.sh` script to admins, Nagios or whatever
- Anyway, the node won’t be used by pgpool-II... Yes, thats problematic if it is the MASTER node that is down
Problems and solutions
Failover’s darkest moment

Problem
- During the slave’s weakening up, there’s no system to lock anything. So write queries could be sent to a read-only backend

Solution
- Known problem, devs currently working on it:

  The obvious idea would be add a config directive which allow pgpool to wait N seconds before the new primary server is ready for accepting connections from clients. More smart idea would be that pgpool checks new primary is available or not before accepting new connections. Better idea? – Tatsuo Ishii SRA OSS, Inc. Japan
Problems and solutions
Failover in a complex architecture with multiple slaves

Problem

- If a failover of the MASTER is done on the PRIMARY, and you have many slaves, only the NEW MASTER (one of the slaves) will be up-to-date
- Because all the slaves are fed by the SR from the former MASTER
- You could then pass from an architecture with MASTER + 4 SLAVES to "ONE MASTER ONLY" architecture: ouch

Solution

- We need CASCADING streaming replication, where a slave can also feed another slave.
- Any PostgreSQL Hacker in the room to state on this ? :-)
- New per Simon’s talk about SR yesterday: We shall all focus on the "Replication manager" tool (repmgr)¹

¹http://projects.2ndquadrant.com/repmgr
Conclusion

pgpool-II + PostgreSQL SR/HS : a good compromise

- A good solution for most cases, even on complex architectures with multiple nodes
- Will be better soon with patches on code and documentation
- Reliable, even more with new repmgr (to be tested soon!)
Stay tuned!

Some URLs

- Main project page
  http://pgpool.projects.postgresql.org/

- pgFoundry page
  http://pgfoundry.org/projects/pgpool/

- General mailing-list
  http://pgfoundry.org/mailman/listinfo/pgpool-general

- Hackers mailing-list
  http://pgfoundry.org/mailman/listinfo/pgpool-hackers

- This talk PDF
  check http://www.dalibo.org/ under "Conferences" soon!

- Twitter @jpargudo
  http://www.dalibo.(com|org)
  jean-paul.argudo@dalibo.com
Any question?
Feel free to ask

- There’s no stupid questions
- But my answers shall be :-/
- Sorry guys, I really tried to have 42 slides