

Implementing High Availability with PostgreSQL

Dimitri Fontaine
dimitri@2ndQuadrant.fr

23 May 2013

1 Agenda

- whoami
- Availability, Durability
- Architectures and Replications

2 Isolate Services

- Traffic growth

3 Durability

- Data Durability
- Data Availability

4 Availability

- Services Availability
- Sharding
- Services and Data Availability

5 Conclusion

- PostgreSQL Replication: Looking back, looking forward
- Questions

2ndQuadrant France PostgreSQL Major Contributor

- pgloader, prefix, skytools, debian, ...
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Replication*
- *Partitioning*



Proven Architectures, implemented lots at Hi-Media

Hi-Media (turnover scale: 200 millions €)

3 different activities to get money from the web

- Allopass, HiPay: internet (micro) payment
- Telecom Service
- Advertising

PostgreSQL is the heart of the technical platform

- Business needs compliance
- Capacity to adapt to changes



Proven Architectures, implemented lots at Hi-Media

Hi-Media (turnover scale: 200 millions €)

3 different activities to get money from the web

- Allopass, HiPay: internet (micro) payment
- Telecom Service
- Advertising



PostgreSQL is the heart of the technical platform

- Business needs compliance
- Capacity to adapt to changes

Proven Architectures, implemented lots at Hi-Media

Hi-Media (turnover scale: 200 millions €)

3 different activities to get money from the web

- Allopass, HiPay: internet (micro) payment
- Telecom Service
- Advertising

PostgreSQL is the heart of the technical platform

- Business needs compliance
- Capacity to adapt to changes



Proven Architectures, implemented lots at Hi-Media

Hi-Media (turnover scale: 200 millions €)

3 different activities to get money from the web

- Allopass, HiPay: internet (micro) payment
- Telecom Service
- Advertising

PostgreSQL is the heart of the technical platform

- Business needs compliance
- Capacity to adapt to changes



PostgreSQL: Your data is our job

How to ensure *both durability* and *availability* of your data?

Usual needs:

- Reliability
- Stability
- Performances
- Growth capacity (think commercial success)
- Continuity and Innovation



1 Agenda

- whoami
- **Availability, Durability**
- Architectures and Replications

2 Isolate Services

- Traffic growth

3 Durability

- Data Durability
- Data Availability

4 Availability

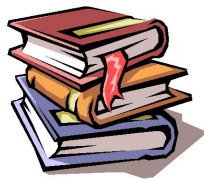
- Services Availability
- Sharding
- Services and Data Availability

5 Conclusion

- PostgreSQL Replication: Looking back, looking forward
- Questions

Some vocabulary

- Availability
- Durability (ACID)
- Architectures
- Replications



Some vocabulary

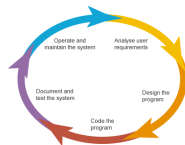
- Availability of services or of data?
- Durability (ACID)
- Architectures
- Replications



Needs first

Needs evolve, solutions must adapt

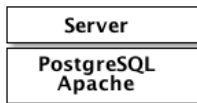
- Start simple
- Some first classic steps
- High Availability of Data
- High Availability of Services
- Read Only Load Balancing
- Read Write Load Balancing



Let's start simple

Our projet life cycle

Let's start with the example of a quite simple project released as a web application seeing its needs evolve with its success.



- 1 Agenda
 - whoami
 - Availability, Durability
 - Architectures and Replications
- 2 Isolate Services
 - Traffic growth
- 3 Durability
 - Data Durability
 - Data Availability
- 4 Availability
 - Services Availability
 - Sharding
 - Services and Data Availability
- 5 Conclusion
 - PostgreSQL Replication: Looking back, looking forward
 - Questions

Scaling out 101

Services Availability

- Front servers are *stateless*
- Watch out for `max_connections`
- Don't you use persistent connections!
- `pgbouncer`



Scaling out 101

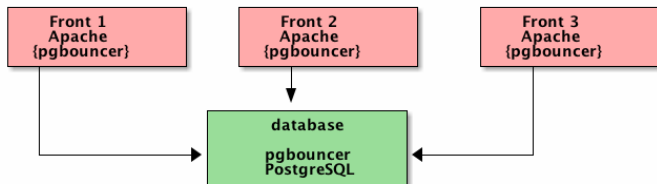
Services Availability

- Front servers are *stateless*
- Watch out for `max_connections`
- Don't you use persistent connections!
- `pgbouncer`

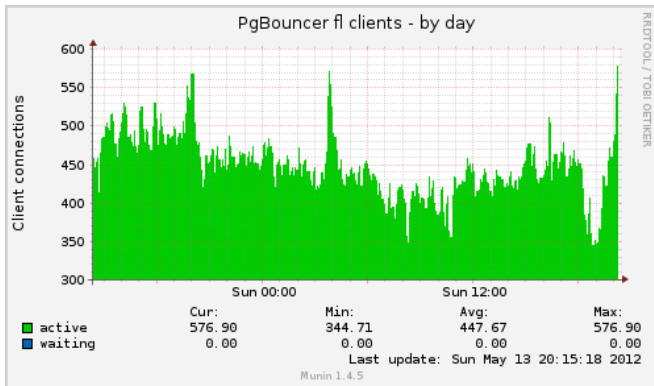


Scaling out 101

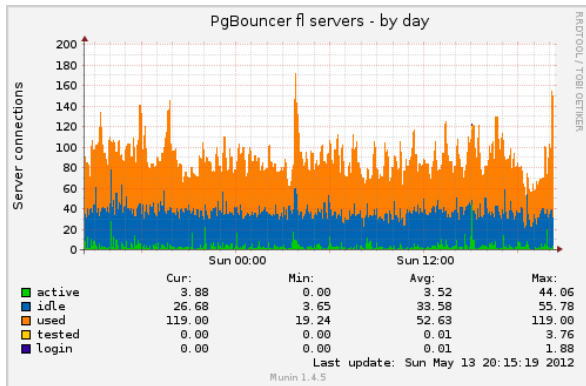
Using more than a single server and a connection pool



pgbouncer is able to reuse client **and** server side connections.



pgbouncer is able to reuse client **and** server side connections.



- 1 **Agenda**
 - whoami
 - Availability, Durability
 - Architectures and Replications
- 2 **Isolate Services**
 - Traffic growth
- 3 **Durability**
 - **Data Durability**
 - Data Availability
- 4 **Availability**
 - Services Availability
 - Sharding
 - Services and Data Availability
- 5 **Conclusion**
 - PostgreSQL Replication: Looking back, looking forward
 - Questions

Getting serious: backups

Backup Strategy is the single most important step towards data availability

- Nightly `pg_dump -Fc`
- Don't forget `pg_dumpall -globals-only`
- Data Retention
- 7 days of nightly backups
- 7 weeks of weekly backups
- 12 months of monthly backups
- 30 years of yearly backups?



Getting serious: backups

Backup Strategy is the single most important step towards data availability

- Nightly `pg_dump -Fc`
- Don't forget `pg_dumpall -globals-only`
- Data Retention
- 7 days of nightly backups
- 7 weeks of weekly backups
- 12 months of monthly backups
- 30 years of yearly backups?



Getting serious: backups

Backup Strategy is the single most important step towards data availability

- Nightly `pg_dump -Fc`
- Don't forget `pg_dumpall -globals-only`
- Data Retention
- 7 days of nightly backups
- 7 weeks of weekly backups
- 12 months of monthly backups
- 30 years of yearly backups?



Failover, 101

`pg_dump`, `pg_restore`

- protection against *errors and omissions*
- beware of restoring time
- still a must have for data **durability**
- what about data **availability**?



Failover, 101

`pg_dump`, `pg_restore`

- protection against *errors and omissions*
- beware of restoring time
- still a must have for data **durability**
- what about data **availability**?



- 1 Agenda
 - whoami
 - Availability, Durability
 - Architectures and Replications

- 2 Isolate Services
 - Traffic growth

- 3 Durability
 - Data Durability
 - Data Availability

- 4 Availability
 - Services Availability
 - Sharding
 - Services and Data Availability

- 5 Conclusion
 - PostgreSQL Replication: Looking back, looking forward
 - Questions

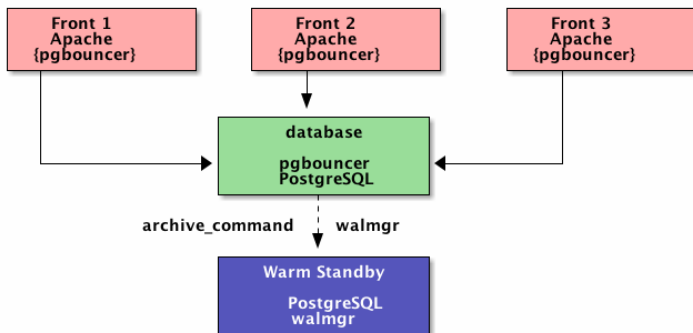
Using physical backups and *Point In Time Recovery*

- **Point In Time Recovery**, 8.1
- *warm standby*, 8.2
- Archiving and *crash recovery*
- `archive_command`
- `restore_command`
- `walmgr.py`, WAL-E



Warm Standby

Implementing *Warm Standby*



1 Agenda

- whoami
- Availability, Durability
- Architectures and Replications

2 Isolate Services

- Traffic growth

3 Durability

- Data Durability
- Data Availability

4 Availability

- Services Availability
- Sharding
- Services and Data Availability

5 Conclusion

- PostgreSQL Replication: Looking back, looking forward
- Questions

Application Split

When you have separate *backoffice* and *production* requirements

- Cross replication
- Slony, Londiste, Bucardo
- Specific processing, *batches*
- Off-line processing
- Still *Transactional* processing
- Skytools comes with **PGQ**



Application Split

When you have separate *backoffice* and *production* requirements

- Cross replication
- Slony, **Londiste**, Bucardo
- Specific processing, *batches*
- Off-line processing
- Still *Transactional* processing
- Skytools comes with **PGQ**



Application Split

When you have separate *backoffice* and *production* requirements

- Cross replication
- Slony, **Londiste**, Bucardo
- Specific processing, *batches*
- Off-line processing
- Still *Transactional* processing
- Skytools comes with **PGQ**



Application Split

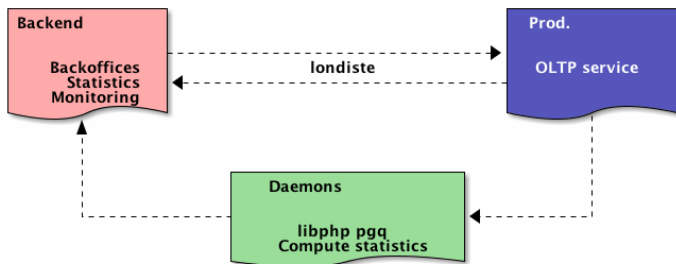
When you have separate *backoffice* and *production* requirements

- Cross replication
- Slony, **Londiste**, Bucardo
- Specific processing, *batches*
- Off-line processing
- Still *Transactional* processing
- Skytools comes with **PGQ**



Application Split

Implementing *londiste* and *PGQ*



Queueing with PGQ

Off-line processing is better done with PGQ

- Mainly written in PLpgSQL (and C)
- Client *API* for python
- and PHP
- some work is happening for Java
- **Cooperative Worker** (Skytools 3)

PGQ: Stable, Reliable, Easy to monitor



Queueing with PGQ

Off-line processing is better done with PGQ

- Mainly written in PLpgSQL (and C)
- Client *API* for python
- and PHP
- some work is happening for Java
- **Cooperative Worker** (Skytools 3)

PGQ: Stable, Reliable, Easy to monitor



Disaster Recovery and Service Continuity

PostgreSQL 9.1 offers *Synchronous Replication* and *Hot Standby*

- Hot Standby
- (A)Synchronous Replication
- Standby connects with `libpq`
- `recovery.conf`
- Continue Archiving
- Switchable per transaction
- Good performance level



Disaster Recovery and Service Continuity

PostgreSQL 9.1 offers *Synchronous Replication* and *Hot Standby*

- **Hot Standby**
- (A)Synchronous Replication
- Standby connects with `libpq`
- `recovery.conf`
- Continue Archiving
- Switchable **per transaction**
- Good performance level



Disaster Recovery and Service Continuity

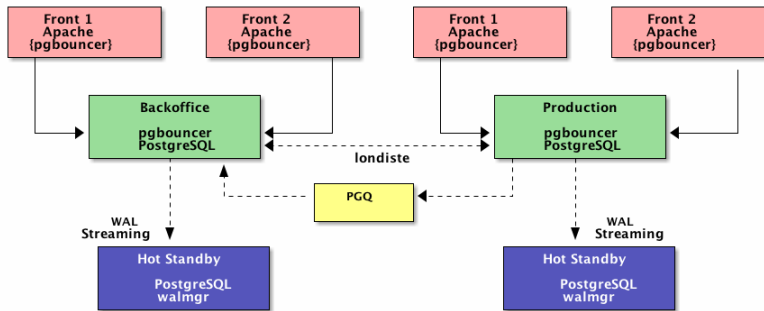
PostgreSQL 9.1 offers *Synchronous Replication* and *Hot Standby*

- **Hot Standby**
- (A)Synchronous Replication
- Standby connects with `libpq`
- `recovery.conf`
- Continue Archiving
- Switchable **per transaction**
- Good performance level



Disaster Recovery and Service Continuity

Implementing Hot Standby



- 1 Agenda
 - whoami
 - Availability, Durability
 - Architectures and Replications
- 2 Isolate Services
 - Traffic growth
- 3 Durability
 - Data Durability
 - Data Availability
- 4 Availability
 - Services Availability
 - Sharding
 - Services and Data Availability
- 5 Conclusion
 - PostgreSQL Replication: Looking back, looking forward
 - Questions

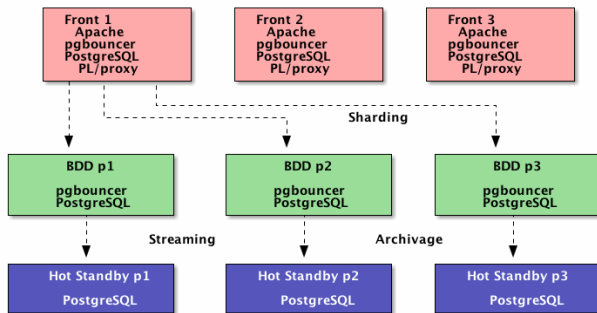
Scaling Writes

PL/proxy

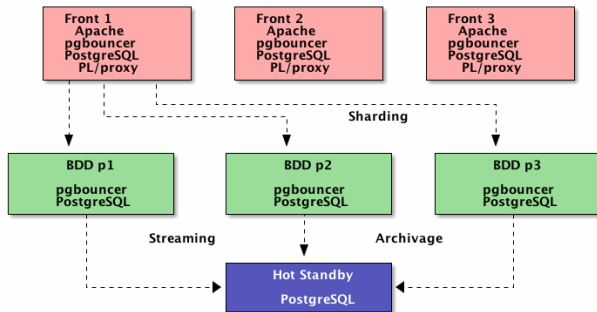
- *Scale-up or Scale-out?*
- *Remote Procedure Call*
- *Sharding*
- Distributed database
- Autonomous Transactions
- **Stored Procedures**



Implementing plproxy



Implementing plproxy



1 Agenda

- whoami
- Availability, Durability
- Architectures and Replications

2 Isolate Services

- Traffic growth

3 Durability

- Data Durability
- Data Availability

4 Availability

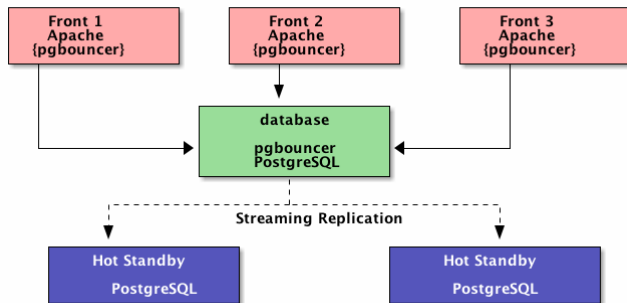
- Services Availability
- Sharding
- Services and Data Availability

5 Conclusion

- PostgreSQL Replication: Looking back, looking forward
- Questions

Services and Data Availability

You need **two** standbys



- 1 Agenda
 - whoami
 - Availability, Durability
 - Architectures and Replications
- 2 Isolate Services
 - Traffic growth
- 3 Durability
 - Data Durability
 - Data Availability
- 4 Availability
 - Services Availability
 - Sharding
 - Services and Data Availability
- 5 Conclusion
 - PostgreSQL Replication: Looking back, looking forward
 - Questions

Retrospective and Future of PostgreSQL Replication

- **8.1, PITR**
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- **8.1, PITR**
- **8.2, Warm Standby**
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- 8.1, PITR
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- 8.1, PITR
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- 8.1, PITR
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- 8.1, PITR
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Retrospective and Future of PostgreSQL Replication

- 8.1, PITR
- 8.2, Warm Standby
- 8.3, `pg_standby`
- 9.0, Hot Standby
- 9.1, Synchronous Replication
- 9.2, Cascading Replication
- 9.3, Event Triggers, *Refactoring*
- 9.4, **Bi-Directional Replication**



Questions?

Now is the time to ask!

