Desmistificando Replicação no PostgreSQL

Euler Taveira

São Paulo, 06/05/2017

Sobre esta apresentação

Timbira
A empresa brasileira de PostgreSQL

- esta apresentação está disponível em:
  `http://www.timbira.com.br/material`
- esta apresentação está sob licença *Creative Commons Atribuição-Não Comercial 3.0 Brasil*:
  `http://creativecommons.org/licenses/by-nc/3.0/br`

Apresentação

- **Euler Taveira**
  - Desenvolvedor PostgreSQL
  - Líder do PostgreSQL Brasil
  - @eulerto
  - `http://eulerto.blogspot.com`
- **Timbira**
  - Diretor Técnico
  - A empresa brasileira de PostgreSQL
  - Consultoria
  - Desenvolvimento
  - Suporte 24x7
  - Treinamento

Resumo

O que é?

- perguntas mais frequentes
- curiosidades
- conceitos de bancos de dados
- como fazer

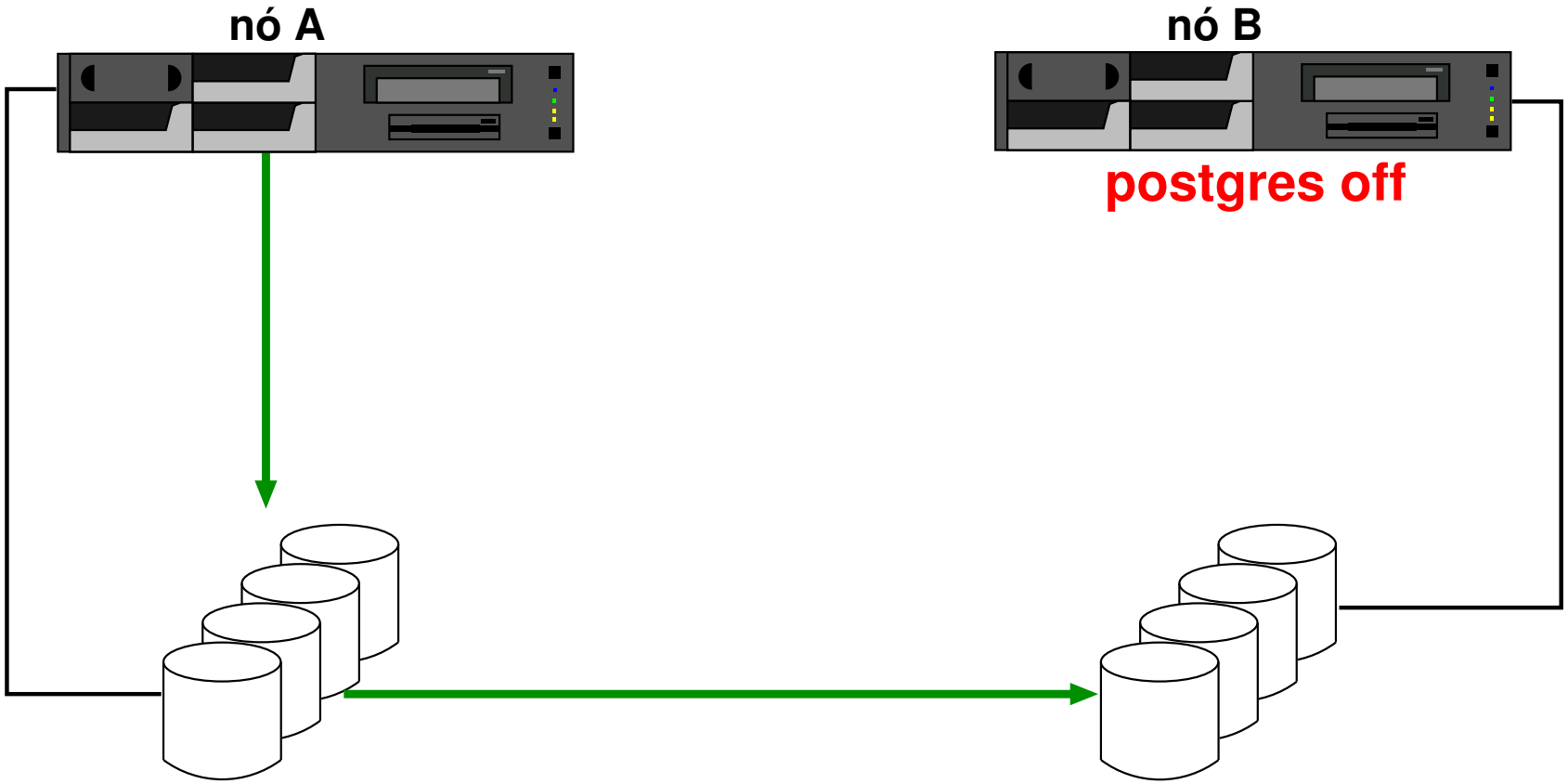
O que não é?

- tópicos avançados
- comparação com soluções de outros SGBDs
- soluções de replicação a nível de sistema de arquivos
- soluções de replicação a nível de hardware

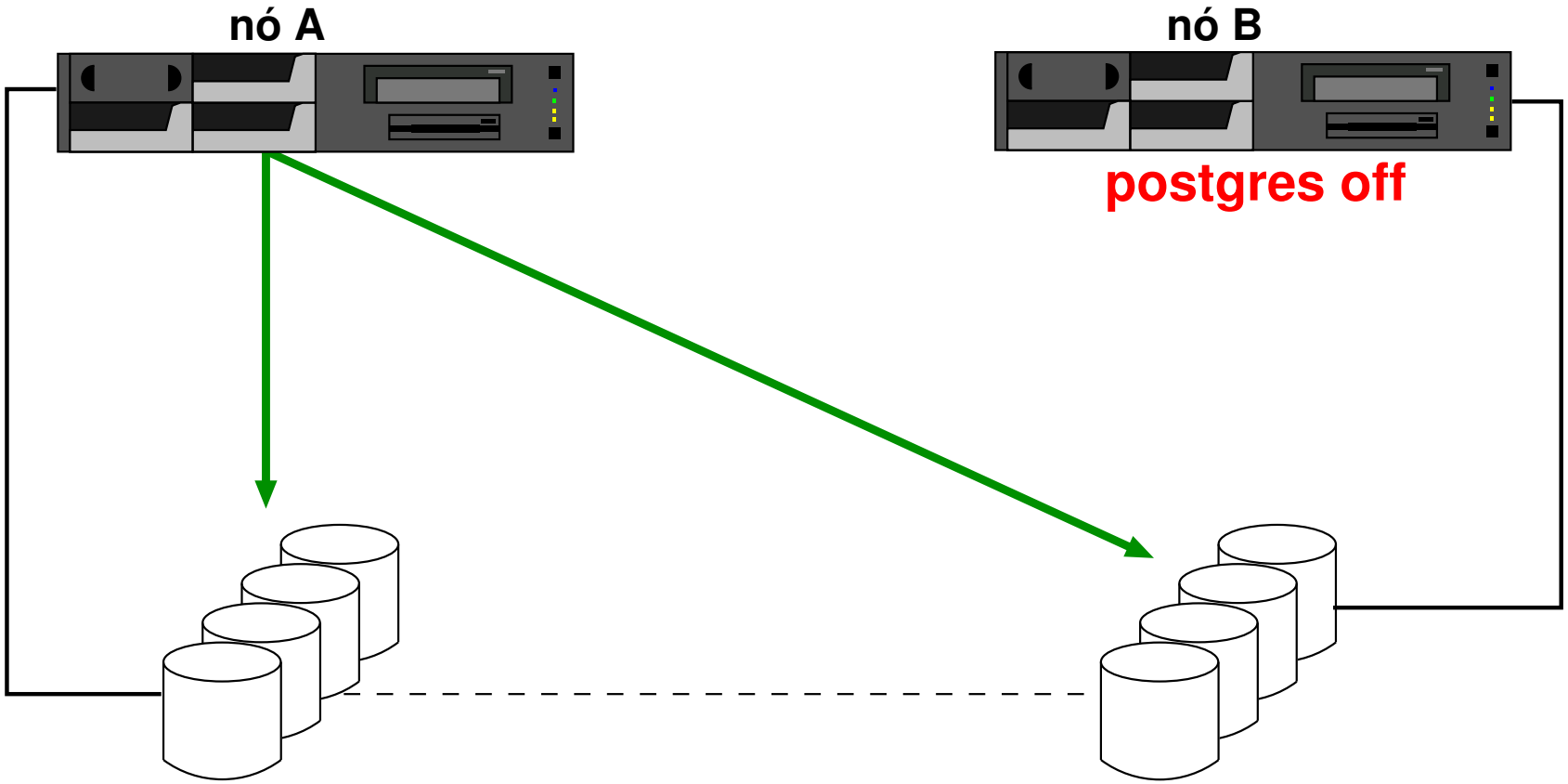
Um pouco de teoria...

- "**Replicação** significa que nós armazenamos várias cópias de uma relação ou partições dela em sites diferentes."
- Motivação:
  - aumentar a disponibilidade
    - problema na réplica
    - falha de comunicação
  - acelerar execução de uma consulta
    - réplica mais próxima pode executar consulta mais rápido
  - balancear a carga no SGBD
  - tolerância a falhas (**SPOF**)
- Como manter a réplica quando a relação é modificada?
  - síncrono
  - assíncrono

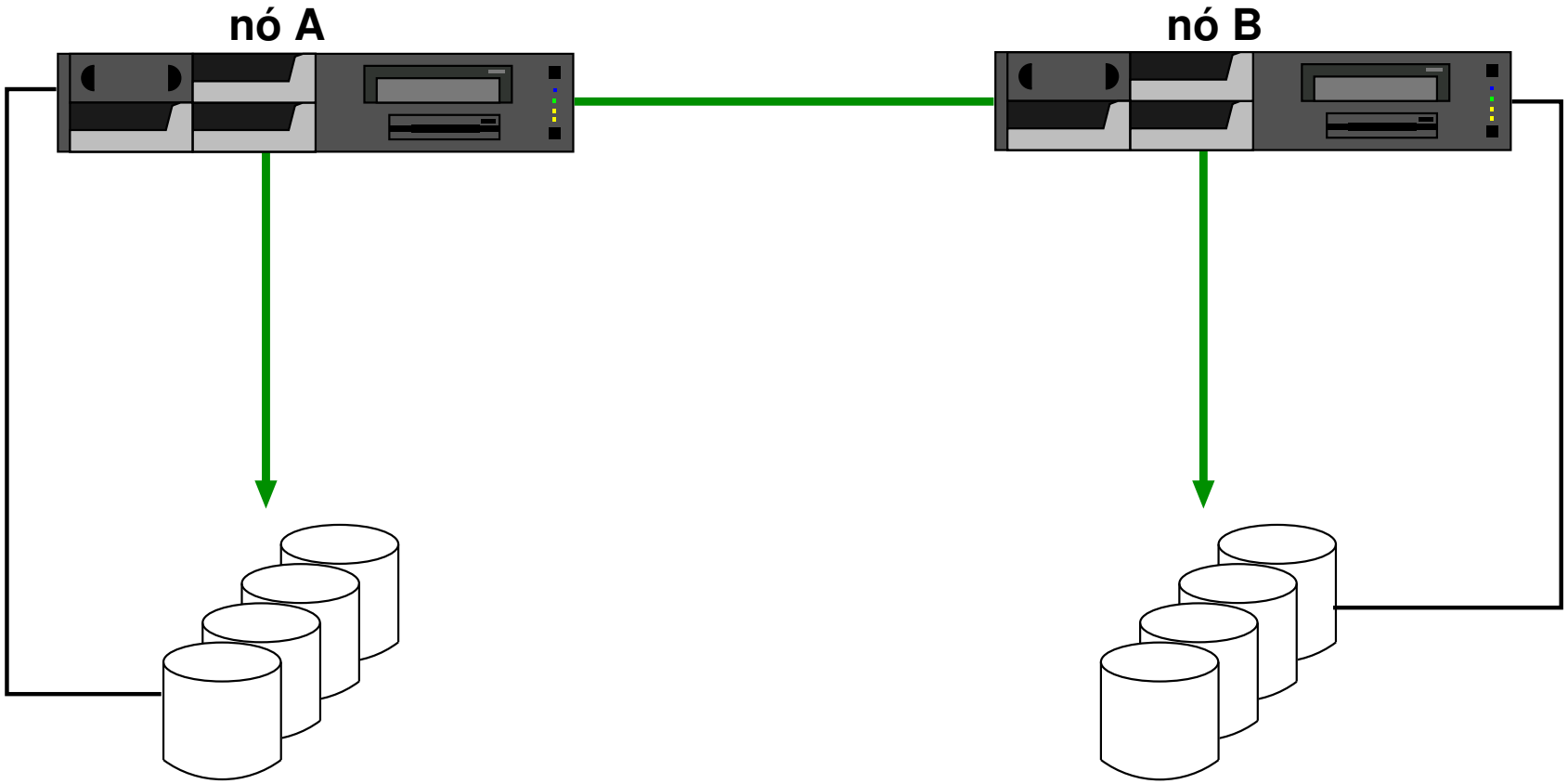
Replicação Física: Hardware



Replicação Física: Sistema Operacional

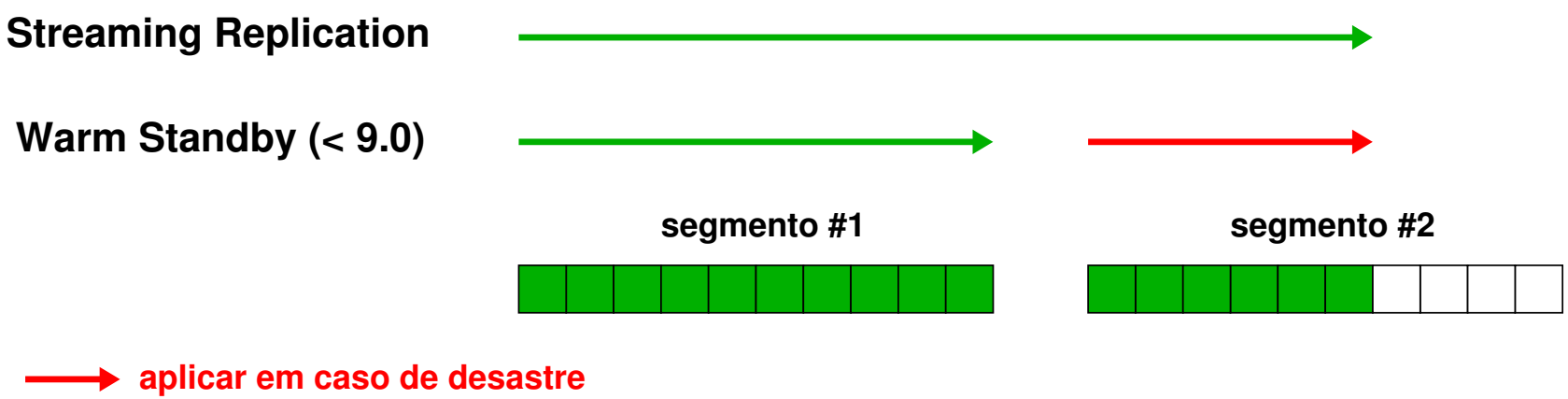


Replicação Lógica



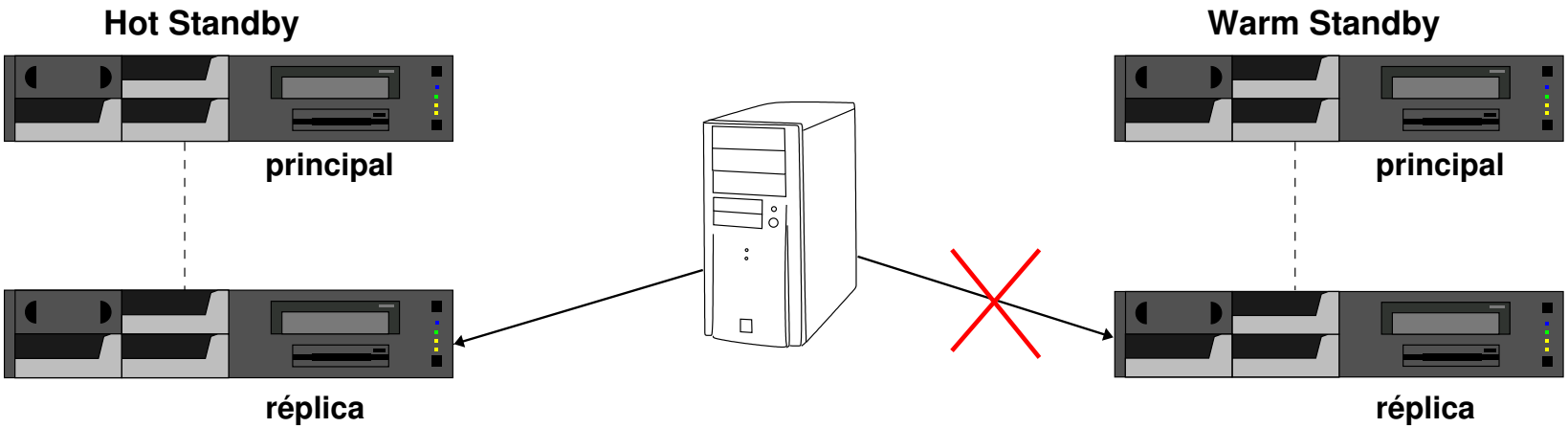
Granularidade

- **segmento de log de transação**: quando um arquivo de log de transação é arquivado, ele é aplicado no outro nó
  - *archive_timeout (longo)*
- **buffer de log de transação**: quando a transação é efetivada, ela é transmitida e efetivada no outro nó
  - $\precsim$ *1 seg (curto)*



Uso do servidor réplica

- **warm standby**: o servidor réplica **não** aceita conexões
- **hot standby**: o servidor réplica aceita conexões



Alta Disponibilidade

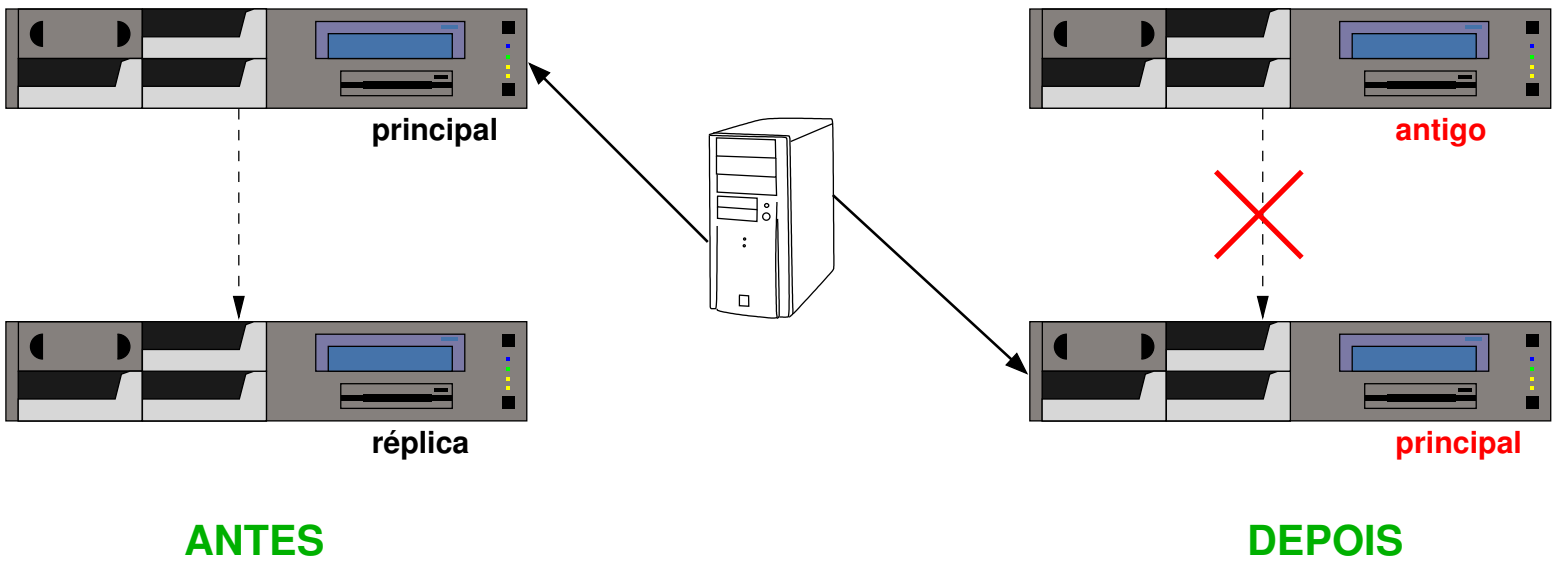
- manter o serviço disponível o máximo de tempo possível
- parada
  - programada (manutenção)
  - não programada (falha / desastre)
- Acordo de Nível de Serviço (**SLA**)
  - porcentagem do *uptime* / tempo
- tempo médio para recuperação
- tempo médio entre falhas

Alta Disponibilidade

Disponibilidade	Parada por ano	Parada por mês
90%	36,5 dias	72 horas
99%	3,65 dias	7,2 horas
99,9%	8,76 horas	43,8 minutos
99,99%	52,56 minutos	4,32 minutos
99,999%	5,26 minutos	25,9 segundos
99,9999%	31,5 segundos	2,59 segundos
99,99999%	3,15 segundos	0,259 segundos

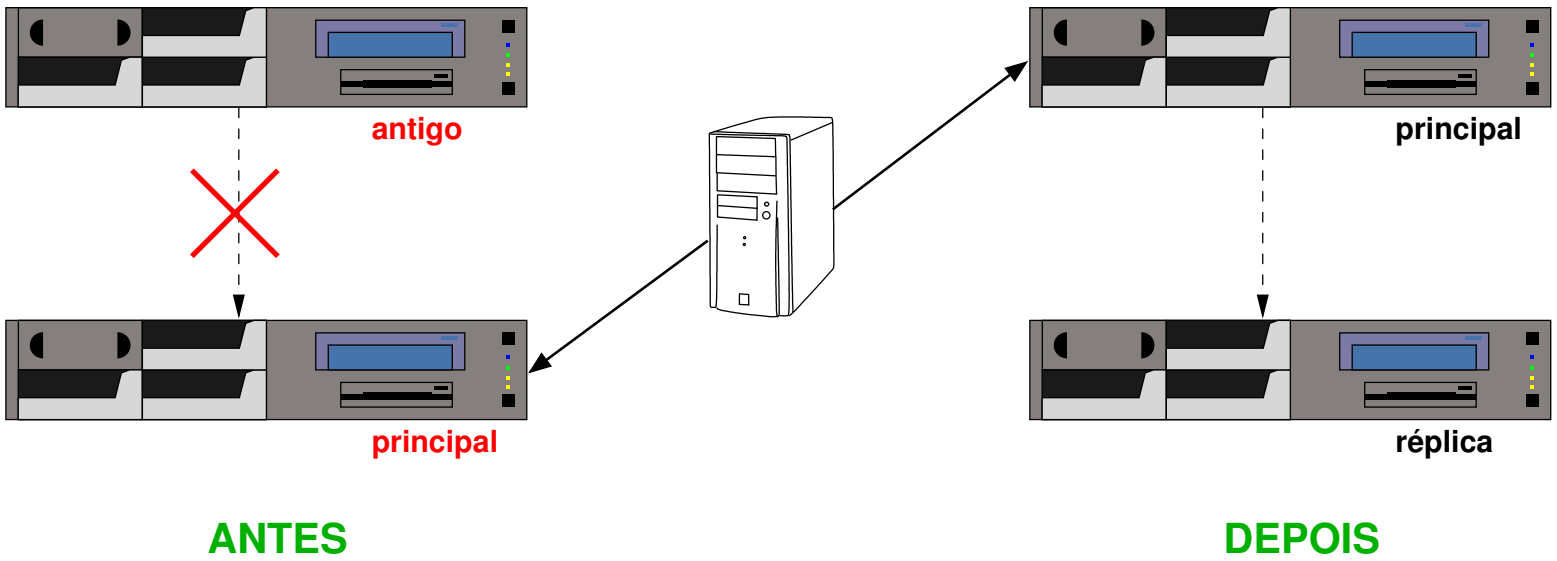
Failover

- transferência do serviço em caso de falha
- quando um servidor falha, outro servidor assume o seu serviço



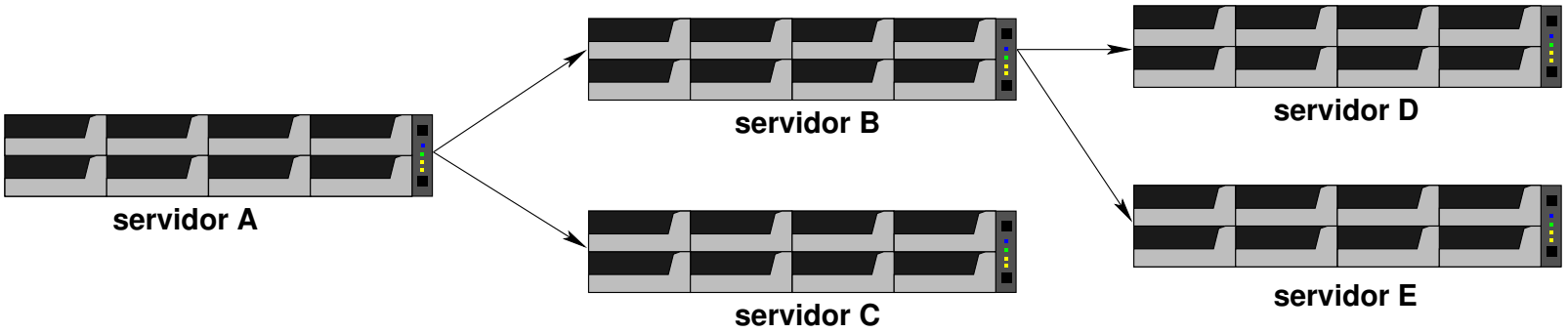
Failback

- retornar serviço ao servidor principal
- estado anterior a falha



Cascateamento

- servidor A replica para servidor B e C
- servidor B replica para servidor D e E



Balanceamento de Carga

- distribuir a carga entre diversos servidores
- algoritmos de agendamento
  - randômico
  - *round robin*
- carga assimétrica
- otimizar a utilização de recursos
- maximizar o desempenho
- evitar sobrecarga

Resumo

Evolução

- **8.0**
  - *warm standby*
- **8.1**
  - *warm standby* (melhorias)
- **9.0**
  - replicação assíncrona
  - *hot standby*
  - protocolo de replicação
- **9.1**
  - replicação síncrona
  - protocolo de replicação (melhorias)
- **9.2**
  - replicação síncrona (*remote_write*)
  - cascateamento
  - cópia base a partir do servidor réplica

Evolução

- **9.3**
  - seguir mudança de *timeline*
  - gatilhos de eventos
  - *background workers*
- **9.4**
  - *slots* de replicação
  - *logical decoding*
  - atraso configurável no servidor réplica
- **9.5**
  - acompanhar progresso da replicação lógica
  - compressão do WAL
  - monitoramento de *slots* de replicação
- **9.6**
  - múltiplos servidores síncronos
  - balanceamento de leitura confiável (*remote_apply*)
- **10.0**
  - replicação lógica
  - facilitar configuração de replicação (parâmetros padrão)

Planejamento

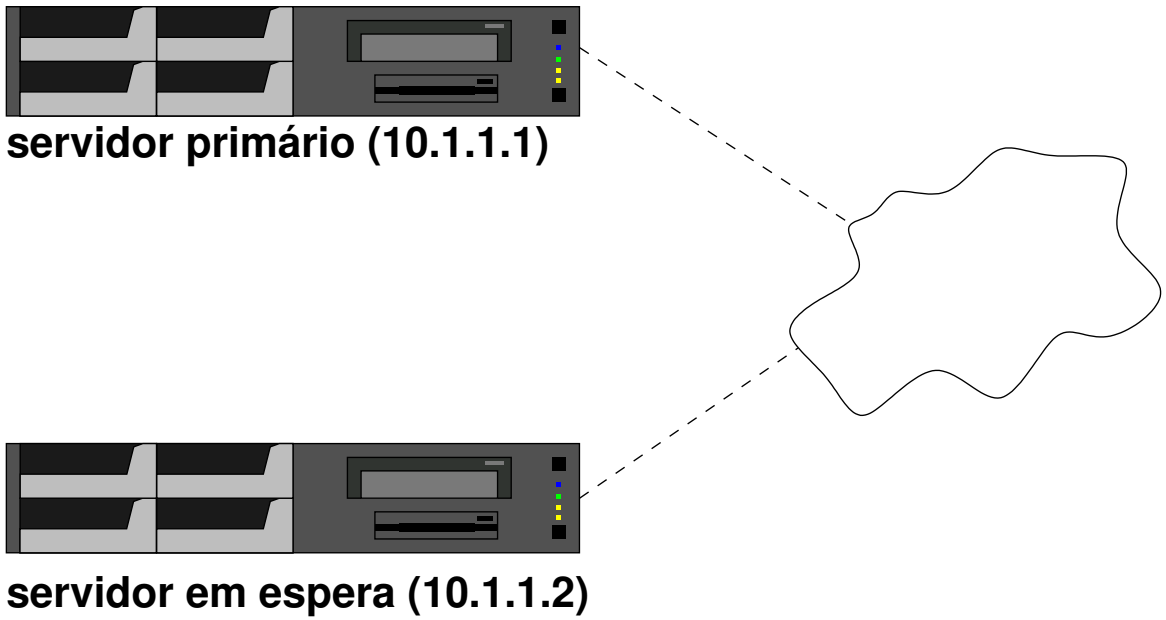
- mesma arquitetura
  - 32 x 64 bits
- mesmo sistema operacional
- mesma versão do PostgreSQL
  - **9.4**.5 $\rightarrow$ **9.4**.11 (funciona)
  - **9.4**.9 $\rightarrow$ **9.5**.6 (**não** funciona)
- mesmos caminhos para *tablespaces*
  - criar caminho nas réplicas **antes** de criar a *tablespace* no servidor principal

Funcionamento

- cópia base: cópia de todo *cluster* para servidor réplica
- recuperação de registros do log de transação no servidor réplica
- entrega
  - arquivos
  - fluxo (*stream*)
    - *walsender* (servidor principal)
    - *walreceiver* (servidor réplica)
- *role*
  - somente fluxo
  - privilégio REPLICATION ($\geq$ 9.1)
- configuração
  - *postgresql.conf*
  - *recovery.conf*

Replicação por Arquivos: Arquitetura

- habilitar arquivamento para repositório
- realizar cópia base do servidor principal
- iniciar restauração contínua no servidor réplica



Replicação por Arquivos

- **restore_command**: script ou software que espera indefinidamente arquivo WAL
  - **pg_standby** ($\geq$ 8.3)

```
1 triggered = false;
2 while (!NextWALFileReady() && !triggered)
3 {
4   sleep(100000L);    /* wait for ~0.1 sec */
5   if (CheckForExternalTrigger())
6     triggered = true;
7 }
8 if (!triggered)
9   CopyWALFileForRecovery();
```

Replicação por Arquivos: No Principal

Timbira
A empresa brasileira de PostgreSQL

postgresql.conf

```
wal_level = replica # hot_standby se < 9.6
archive_mode = on
archive_command = 'scp %p usuario@10.1.1.2:/archives/%f'
```

Cópia Física: Serviço Parado

```
$ pg_ctl stop -D /bd/primario
waiting for server to shut down.... done
server stopped
$ rsync -av --exclude postgresql.conf \
--exclude pg_hba.conf --exclude pg_xlog/* \
--exclude pg_log/* /bd/primario/ \
postgres@10.1.1.2:/bd/secundario
$ pg_ctl start -D /bd/primario
server starting
```

Cópia Física: Serviço em Execução

```
postgres=# select pg_start_backup('replicacao', true);
pg_start_backup
-----------------
0/5044CB4
(1 row)
$ rsync -av --exclude postmaster.pid \
--exclude postgresql.conf --exclude pg_hba.conf \
--exclude pg_xlog/* \ --exclude pg_log/* /bd/primario/ \
postgres@10.1.1.2:/bd/secundario
postgres=# select pg_stop_backup();
pg_stop_backup
---------------
0/90D7950
(1 row)
```

Replicação por Arquivos: Na Réplica

Timbira
A empresa brasileira de PostgreSQL

recovery.conf

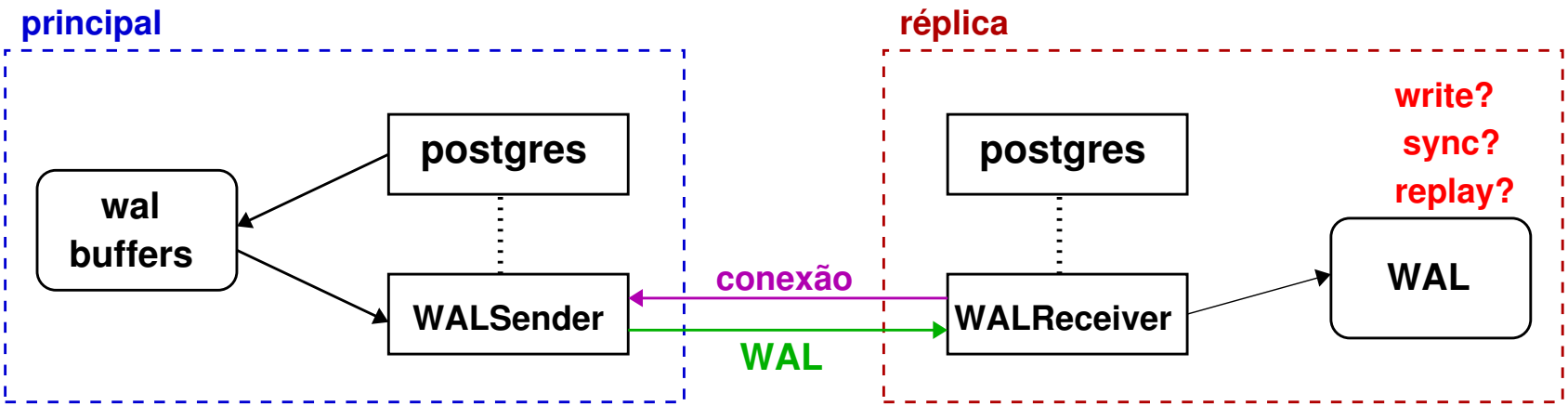
```
restore_command = 'pg_standby -t /tmp/f.trg -d /archives %f
%p %r 2>>/tmp/stdby.log'
archive_cleanup_command = 'pg_archivecleanup /archives %r'
recovery_end_command = 'rm -f /tmp/f.trg'
```

postgresql.conf

```
hot_standby = on
```

Replicação por Fluxo: Arquitetura

- *WALReceiver* estabelece uma conexão (via libpq) com servidor principal
- servidor principal abre o processo *WalSender* para enviar WAL ao servidor réplica
- replicação síncrona espera WAL ser escrito e/ou aplicado no servidor réplica



Replicação por Fluxo: Assíncrona

- replicação por fluxo no PostgreSQL é **assíncrona** por padrão
- se o servidor principal cair, algumas transações que foram efetivadas podem não ter sido replicadas
- a quantidade de dados perdidos é correspondente ao atraso da replicação no momento da queda

Curiosidade

A partir da 9.4, é possível configurar o servidor réplica com atraso predefinido

Replicação por Fluxo: Síncrona

- confirma que todas as mudanças feitas na transação foram transferidas para pelo menos **um** servidor réplica
- cada transação que modifica dados esperará a **confirmação** que as mudanças foram escritas no log de transação de **ambos** servidores
- fornece um nível mais alto de durabilidade

Confiabilidade

A partir da 9.6, você pode exigir a resposta de **n** servidores réplica antes de concluir a transação. *Quorum Commit* está disponível a partir da 10.

Replicação por Fluxo: Síncrona

- tempo da transação
  - transferir os dados entre servidor principal e réplica
  - escrever dados no log de transação do servidor réplica
  - mandar mensagem do servidor réplica para principal com ACK
  - escrever dados no log de transação do servidor principal
- transações somente leitura, ROLLBACK e subtransações **não** esperam resposta do servidor réplica

Replicação em Cascata

- servidor réplica aceita conexões para replicação de outros servidores réplica
- replicação em cascata é **assíncrona**
- **não** há configuração especial para habilitar a replicação em cascata
- se $\geq$ 9.3, capaz de seguir a nova *timeline*

Antes da 9.3

Promover um servidor réplica intermediário termina as conexões de replicação; é necessário refazer as réplicas.

Replicação por Fluxo: No Principal

Timbira
A empresa brasileira de PostgreSQL

postgresql.conf

```
listen_addresses = '*'
wal_level = replica # hot_standby se < 9.6
max_wal_senders = 3
max_replication_slots = 3
wal_keep_segments = 100
synchronous_standby_names = '*'
```

Role de replicação

```
CREATE ROLE usuario LOGIN REPLICATION;
```

pg_hba.conf

```
host replication usuario 10.1.1.2/32 md5
```

Cópia Física: Serviço Parado

```
$ pg_ctl stop -D /bd/primario
waiting for server to shut down.... done
server stopped
$ rsync -av --exclude postgresql.conf \
--exclude pg_hba.conf --exclude pg_xlog/* \
--exclude pg_log/* /bd/primario/ \
postgres@10.1.1.2:/bd/secundario
$ pg_ctl start -D /bd/primario
server starting
```

Cópia Física: Serviço em Execução

```
postgres=# select pg_start_backup('replicacao', true);
pg_start_backup
-----------------
0/5044CB4
(1 row)
$ rsync -av --exclude postmaster.pid \
--exclude postgresql.conf --exclude pg_hba.conf \
--exclude pg_xlog/* \ --exclude pg_log/* /bd/primario/ \
postgres@10.1.1.2:/bd/secundario
postgres=# select pg_stop_backup();
pg_stop_backup
---------------
0/90D7950
(1 row)
```

Cópia Física: pg_basebackup

```
$ pg_basebackup --pgdata=/bd/secundario --verbose \
> --write-recovery-conf --progress \
> --dbname='host=10.1.1.1 port=5432 user=usuario'
```

Replicação por Fluxo: Na Réplica

Timbira
A empresa brasileira de PostgreSQL

recovery.conf

```
standby_mode = 'on'
primary_conninfo = 'host=10.1.1.1 user=usuario password=1234'
trigger_file = '/bd/secundario/failover.trg'
primary_slot_name = 'no1'
recovery_target_timeline = 'latest'
recovery_min_apply_delay = 2h
```

postgresql.conf

```
hot_standby = on
```

Replicação Lógica: No Principal

Timbira
A empresa brasileira de PostgreSQL

postgresql.conf

wal_level = logical

Role de replicação

CREATE ROLE usuario LOGIN;

pg_hba.conf

host foo usuario 10.1.1.2/32 md5

Publicar tabelas

foo=# create publication pub1 for table contas, historico, vendas;
CREATE PUBLICATION

Replicação Lógica: Na Réplica

Restaurar esquema

```
pg_dump -h 10.1.1.1 -s -U usuario foo | psql -f - -U timbira bar
```

Assinar tabelas

```
bar=# create subscription sub1 connection 'host=10.1.1.1
user=usuario dbname=foo' publication pub1;
CREATE SUBSCRIPTION
```

Failover

- no *pg_standby*
  - crie o arquivo especificado pela opção *-t*
- criar arquivo de gatilho (*trigger_file*)
  - só tem efeito com *standby_mode = on*
- executar *pg_ctl promote*

Failback

- operação mais complicada do que *failover*
- servidor antigo pode conter dados que **não** estão presentes no **novo servidor principal**
  - não há como desfazer essas transações "perdidas" (não há log de UNDO)
- a solução é:
  - descartar dados do servidor antigo (considerar transações "perdidas"?)
  - montar replicação com o servidor antigo sendo a réplica do novo servidor principal
  - bloquear acesso ao PostgreSQL para promover servidor
  - promover a réplica (servidor antigo)
  - descartar dados do novo servidor (antiga réplica)
  - montar replicação com cenário inicial
- o *pg_rewind* pode ajudar nessa tarefa

Monitoramento

- monitoramento
  - pg_stat_replication ($\geq$ 9.1) – principal
  - pg_stat_database_conflicts ($\geq$ 9.1) – réplica
  - pg_stat_wal_receiver ($\geq$ 9.6) – réplica
  - pg_replication_slots ($\geq$ 9.4)
  - pg_current_xlog_location (principal) e pg_last_xlog_{receive, replay}_location (réplica)

Monitoramento: No Principal

```
postgres=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
 pid              | 7466
 usesysid         | 10
 usename          | replicacao
 application_name | walreceiver
 client_addr      | 10.1.1.2
 client_hostname  |
 client_port      | 51981
 backend_start    | 2014-07-29 21:54:43.573871-03
 backend_xmin     |
 state            | streaming
 sent_location    | 0/16ACA50
 write_location   | 0/16ACA50
 flush_location   | 0/16ACA50
 replay_location  | 0/16ACA50
 sync_priority    | 0
 sync_state       | async
```

Atraso da Replicação

```
postgres=# SELECT pg_size_pretty(
pg_xlog_location_diff(sent_location, replay_location))
as replication_lag FROM pg_stat_replication;
 replication_lag
-----------------
 40 kB
(1 registro)
```

Até a 9.1

Não havia a função pg_xlog_location_diff.

Resumo

Algumas ferramentas...

- ~~rserv~~
- Slony-I
- Londiste
- Bucardo
- pgpool-II
- ~~PGCluster~~
- pglogical
- BDR
- Postgres-XC $\rightarrow$ Postgres-X2
- Postgres-XL

BDR

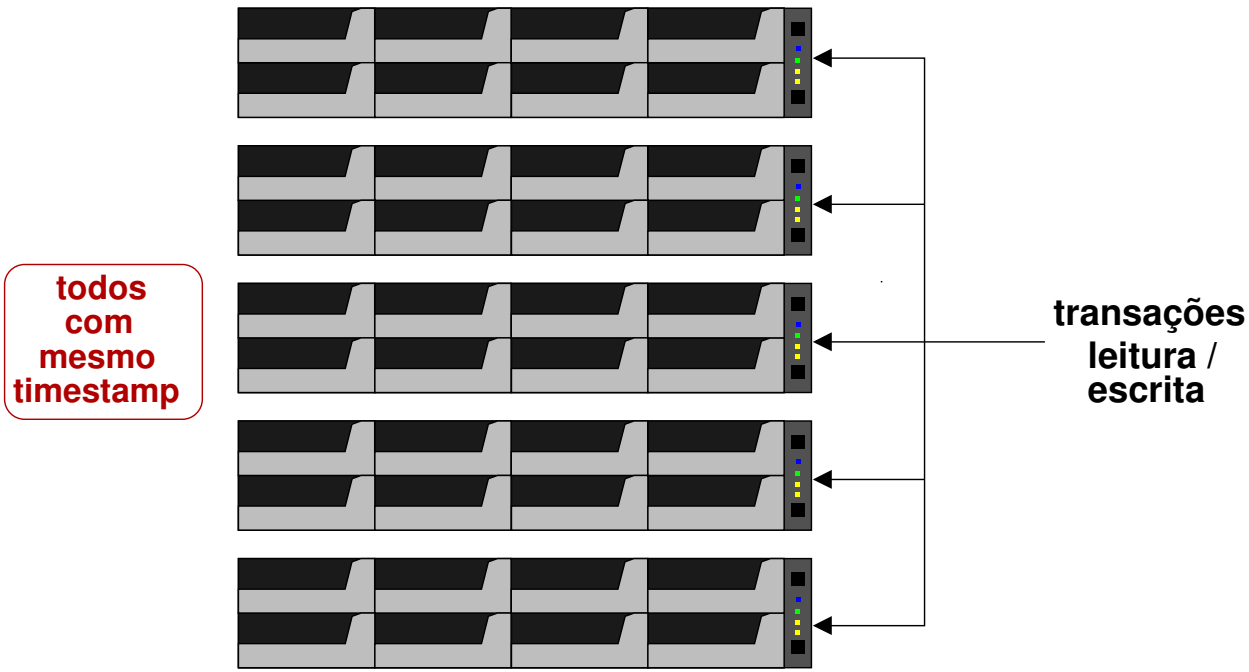
- *Bi-Directional Replication*
- extensão do PostgreSQL
- incorporando funcionalidades aos poucos no repositório do PostgreSQL
- suporta ~~9.3~~ e 9.4
- é necessário uma versão modificada do PostgreSQL
  - o UDR pode rodar no 9.4 sem modificações

BDR: Comparativo

	BDR	HS	Londiste	Slony	Bucardo
Multi-master	sim	não	não ⋆	não	sim
Por Banco	sim	não	sim	sim	sim
Cascateamento	não	sim	sim	sim	-
DDL	sim	sim	não ⋆	não ⋆	não
Daemon externo	não	não	sim	sim	sim
Novas Tabelas Adicionadas	sim	sim	não	não	não
Sequências Transparentes	sim	-	-	-	não
Usa gatilhos / Escrita 2x	não	não	sim	sim	sim
UPDATE na PK	sim	sim	não	não	não
Replicação Seletiva	sim	não	sim	sim	sim
Aplica Txn Individual	sim	sim	não	não	não

extraído do site do BDR

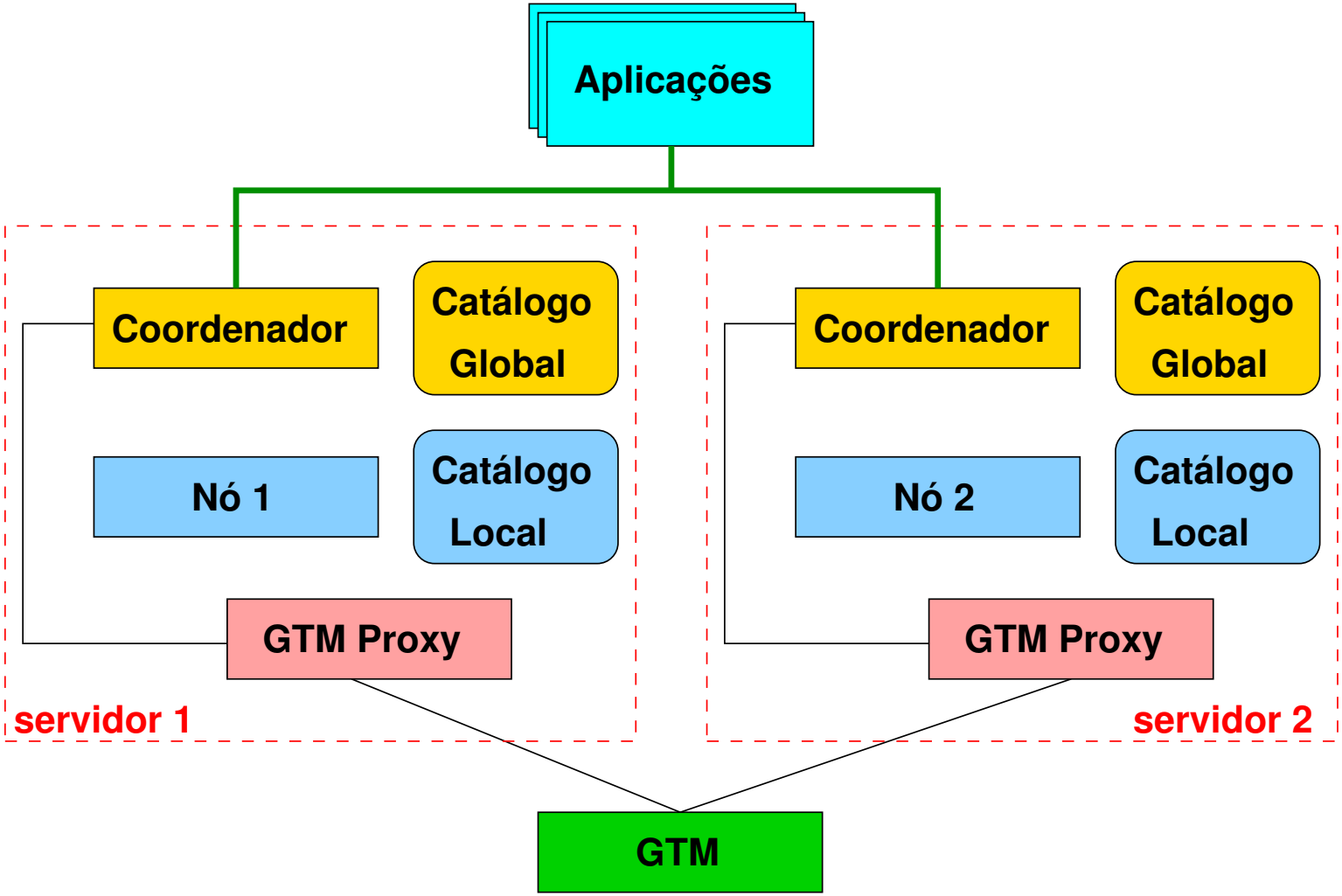
Postgres-XL



Postgres-XL

- arquitetura *shared nothing*
- multi-mestre síncrono
- escalável em leitura/escrita
  - "3,4x performance com 5 servidores comparado com um servidor PostgreSQL"
- local de tabelas transparente
  - tabelas replicadas
  - tabelas distribuídas
- baseado no PostgreSQL (atualmente 9.5)
- mesma API para aplicações que já utilizam PostgreSQL

Postgres-XL: Arquitetura



Resumo

Outras inúmeras perguntas...

- A sua pergunta na lista pgbr-geral
- A sua pergunta na lista pgsql-{general, performance, admin}
- histórico das listas
- blogs
  - `http://planeta.postgresql.org.br`
  - `http://planet.postgresql.org`
- wiki
  - `http://wiki.postgresql.org`
- IRC
  - irc.freenode.net
  - #postgresql
  - #postgresql-br

Referências

- **BDR**: http://www.bdr-project.org/
- **Bucardo**: http://www.bucardo.org/
- **Londiste**: https://wiki.postgresql.org/wiki/SkyTools
- **Slony-I**: http://www.slony.info/
- **Postgres-XL**: https://www.2ndquadrant.com/en/resources/postgres-xl/
- **pgpool-II**: http://www.pgpool.net/

Perguntas

?

Euler Taveira de Oliveira
euler@timbira.com.br
http://www.timbira.com.br