

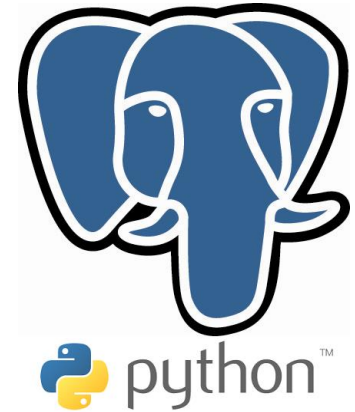
Apresentando PyReplica

Sistema de replicação
simples e personalizável
baseado em Python para PostgreSQL

Mariano Reingart (ArPUG/PyAr)

Emanuel C. Franco (ArPUG/AOSUG)

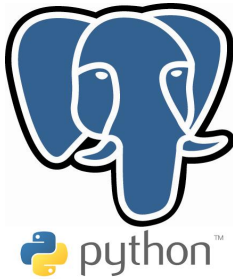
PgCon Brasil 2009



Palestras Traduzido (English & Español):

<http://www.sistemasagiles.com.ar/trac/wiki/PyReplica>

PyReplica: Motivação / Objetivos



- Fácil instalação (scripts, sem compilação)
- Fácil administração (sem comandos!)
- Fácil adaptação (filtro / transformar)
- Fácil manutenção (código *KISS*)
- Eficiente (memória, rede, sem *polling*)
- Multiplataforma (Windows / Linux [Debian default])

Alternativas: PgPool-II



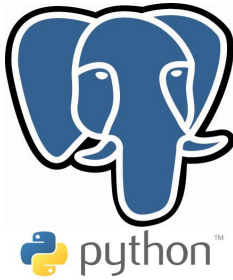
Homem do meio entre PostgreSQL e aplicações:

- Conexão exterior
- Sincronização de replicação
- Balanceamento de carga
- Consultas paralelas

Possíveis desvantagens:

- Ponto único de falha
- Problemas com algumas funções (sequências, aleatórios, CURRENT_TIMESTAMP, etc)
- Suporte limitado de autenticação, COPY, etc
- Degenera / pára se a sincronização está perdida

Alternativas: Slony-I



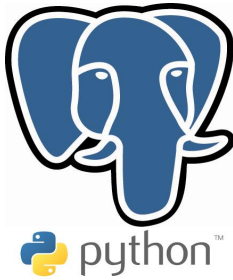
Master / Slave replicação usando disparadores:

- Desenhado para datacenters / sites backups (todos os nós geralmente disponíveis, a rede de confiança)
- Advanced esquema de configuração
- "Little-língua" (comandos) para a gestão

Possíveis desvantagens:

- Complexo para instalar, dar manutenção e configurar
- Somente um mestre (por escrito)
- Inviável para as redes instáveis ou configurações variáveis
- Não replica DDL, LOs, Serials - nem o pyreplica : (

Alternativas: PgCluster / CyberCluster



Multimaster Sync replicator (servidor "especial"):

- Funcionalidade avançada
- Trabalha com funções, LOs, sequências, etc
- Sincronização de alta performance
- O balanceamento de carga / acompanhamento

Possíveis desvantagens:

- Requer servidor PostgreSQL "remendado" ("patched")
- Complexo para instalar e configurar
- Exige muito do hardware

Então, por que não usar ...



PgPool SkyTools Slony-I PgReplicator PgCluster

No nosso caso:

- Difícil de programar (C/Perl, compilação, erros, etc)
- Complexo (dezenas de arquivos, milhares LOCs)
- Difícil de gerir (vários comandos, opções, etc)
- Alguns não funciona no Windows
- Instável em condições extremas (poucos recursos, pouco fiáveis rede, etc)

Características PyReplica



- Assíncrono
- Solução síncrona para aplicações em Python
- Master / Slave e Multimaster limitada
- Replicação Condicional
- A detecção de conflitos
- Notificações por e-mail
- Mantém viva a conexão (KeepAlive)

- Ligação direta a backends
- Sem protocolo especial (SQL query strings)
- Sem ferramentas externas (rsync)
- Duas fases (2PC esta versão)

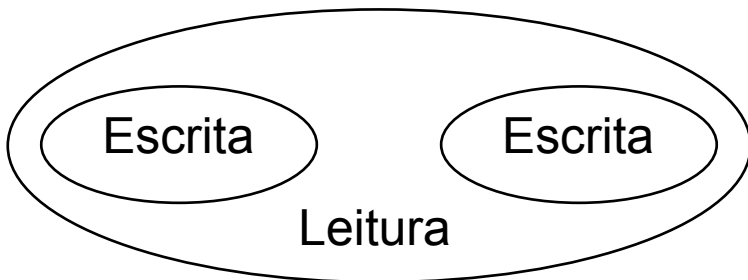
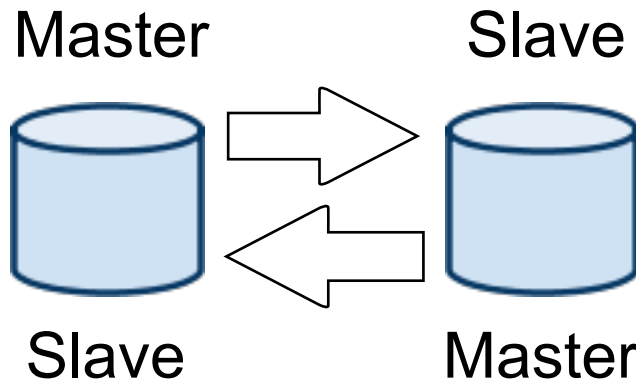
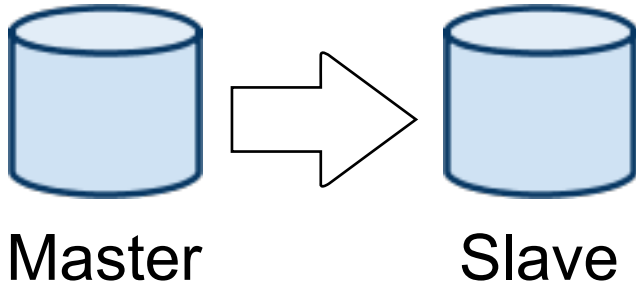
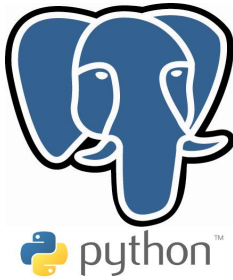
Inconvenientes PyReplica



Como a maioria dos replicadores baseados em gatilhos, PyReplica não suporta:

- Replicação automática de DDL (embora possa ser propagada para vários servidores facilmente)
- LOs replicação (usar em vez bytea)
- Seqüência de replicação (uso começa diferentes!)
- Sincronização de replicação (apenas para python agora)
- Resolução de conflitos (pode ser detectado ou evitadas com regras / triggers)
- Ele suporta: Replicação de valores retornados por data, aleatórias, seqüência, funções, etc

PyReplica: Casos



- Master/Slave: read-only escravos
 - Backup / Failover
 - O balanceamento de carga
 - Datawarehouse / Consolidação
- Multimaster: ambos mestres lêem / escrevem
 - Redundância / alta disponibilidade
 - Servidores remotos (ej. de Ponto de Venda)
 - Servidores Móveis (vendedor ej.)
 - Partição lógica de dados necessária para evitar conflitos de gravação

PyReplica Programming



- Linguagem Python::
 - Compacto, simples, código claro
 - Pilhas incluídas: e-mail, postagens, config ...
- Fácil de programar:
 - 3 arquivos fonte principal, 500 LOC estimadas
 - <150 LOC gatilho e replicador
 - Est. 1 / 8 Slony-I gatilho tamanho (LOC)
 - Desenvolvimento Windows e Linux
 - Sem compilação
 - Suite de testes automatizados

PyReplica structure



py_log_trigger:

- Log gatilho: detecta, reagrupa e armazena consultas SQL para replicação

pyreplica.py:

- Cliente do replicador
- Executa consultas SQL de replicação de escravos

daemon.py:

- Configuração, dispara os threads, notificação de e-mail, keepalives, etc.

Gatilho py_log_replica



- Converte os dados: Python & PostgreSQL
- Detecta alteração, gera SQL:
 - INSERT
 - UPDATE (SELECT para detecção de conflitos)
 - DELETE
- Verifica Condições
- Armazena o SQL na tabela de log (replica_log)
- Notifica réplicas

Replicator (client daemon)



- Conecta-se a dois bancos de dados
- Escuta as notificações
- Executa os SQLs armazenados
- Lida com as transações 2PC
- Lida com Threading (uma para master / slave par)
- Envia e-mails com notificações
- Registra log de texto de auditoria

Instalando PyReplica (1/3)



- Dependências: python, plpython, psycopg2

```
apt-get install postgresql-plpython-8.3 python2.5  
python-psycopg2
```

- Obtenha uma cópia PyReplica (svn melhor opção):

```
svn co http://www.sistemasagiles.com.  
ar/svn/pyreplica  
/usr/local/pyreplica
```

Instalando PyReplica (2/3)



- Banco de dados mestre Dump, restabelecer escravo:

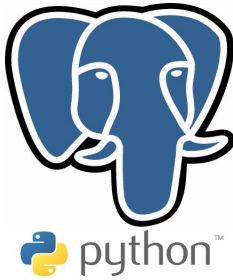
```
createdb somedb -h remote
```

```
pg_dump somedb | psql somedb -h remote
```

- Instalação de replicação no master:

```
psql somedb -U someuser < master-install.sql
```

Instalando PyReplica (3/3)



Configurar e iniciar daemon:

- Copiar e editar arquivo de configuração:

```
cp /usr/local/pyreplica/sample.conf  
  /etc/pyreplica/somedb.conf  
vi /etc/pyreplica/somedb.conf
```

- Iniciar o daemon:

```
/usr/local/pyreplica/daemon.py start
```

Arquivo de configuração



```
[MAIN]
NAME=mydb
# master:
DSN0=dbname=somedb user=someuser password=secret host=remote
# slave:
DSN1=dbname=somedb user=someuser password=secret host=localhost
.
[SMTP]
SERVER=somehost.somewhere.com
START_SUBJECT=[PyReplica] Starting mydb replication
STOP_SUBJECT=[PyReplica] Stopping mydb replication
ERROR_SUBJECT=[PyReplica] Starting mydb replication (ERROR)
WARNING_SUBJECT=[Replica] WARNING on mydb replication
FROM_ADDR=no-reply@somewhere.com
TO_ADDRS=dba@somewhere.com
```

Replicação condicional



py_log_filter:

- relname: nome da tabela a ser filtrada
- event: INSERT, DELETE o UPDATE
- condition: expressão python (True/False)

- Python dicts novos e antigos para comparação

Ej:

- `new["field1"].startswith("Something")`
- `old["field2"]==123`

Testes de desempenho



- Horas estimadas de 100,000 iterações:

```
INSERT INTO test (t,n,f,b) VALUES (random()::text, random(), now(), True);
```

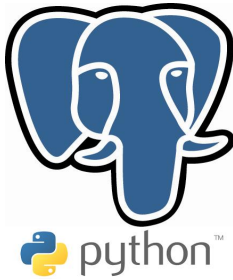
- Sem gatilho: 3,5 s
- Slony-I trigger (C): 8,8 s
- Gatilho dummy de teste (plsql): 11,5 s
- Gatilho PyReplica (plpythonu): 15,1 s

Pendências



- Evitar ou resolver conflitos (multimaster):
 - Nenhuma intervenção manual
 - Evite bloqueios desnecessários
- Melhorar a instalação:
 - Packages (linux). OneClickInstallers (windows)
- Suporte ao daemon on Windows:
 - Corrigir o LISTEN no python-psycopg2 (ou usar outra coisa)
 - Melhor manipulação de fork, e de sinais, etc. (não disponível em windows)
- Melhorar daemon, estender os testes
- txid_snapshot (8.3+) ao invés de 2PC

PyReplica-Admin



Administrador "Visual" para o PyReplica:

- Mais fácil instalação e gestão
- Programado em Python & PythonCard (WX)
- Multiplataforma: Windows, Linux, Mac
- Backup e restauração
- Monitorar o status de replicação
- Suporte Master / Slave e Multimaster
- Fácil edição de normas para a replicação condicional

Obrigado Yanier Gonzalez Perez et. al. de Cuba
(veja [documento com a tese](#))

ALERCE

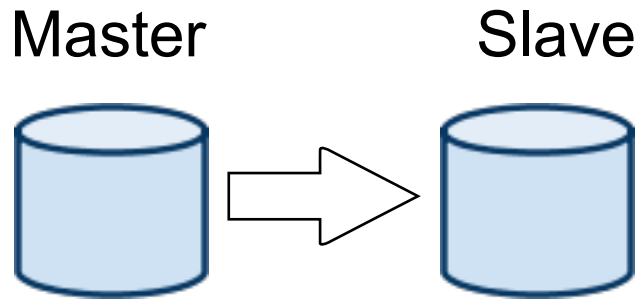
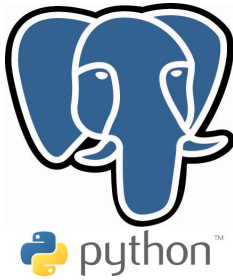


Sync Replicator for Python:

- DbAPI conector compatível (similar Sequoia JDBC)
- DB "Virtual" transparente para o aplicativo
- Master - multiple Splave schema
- Simple Failover - alteração de qualquer escravo (sincronizada!)
- 2PC remotos a bases de dados de sincronização
- pg_log_replica (gatilho) para iniciar a replicação SQL e fácil failback (recuperação de escravo perdido)

Desenvolvimento financiados pela [MSA](#) - Magic Software

Estudo de case: Master/Slave



- Servidor de DNS simples
- Async
- Evite as transferências de zona
- Backup se mestre falhar

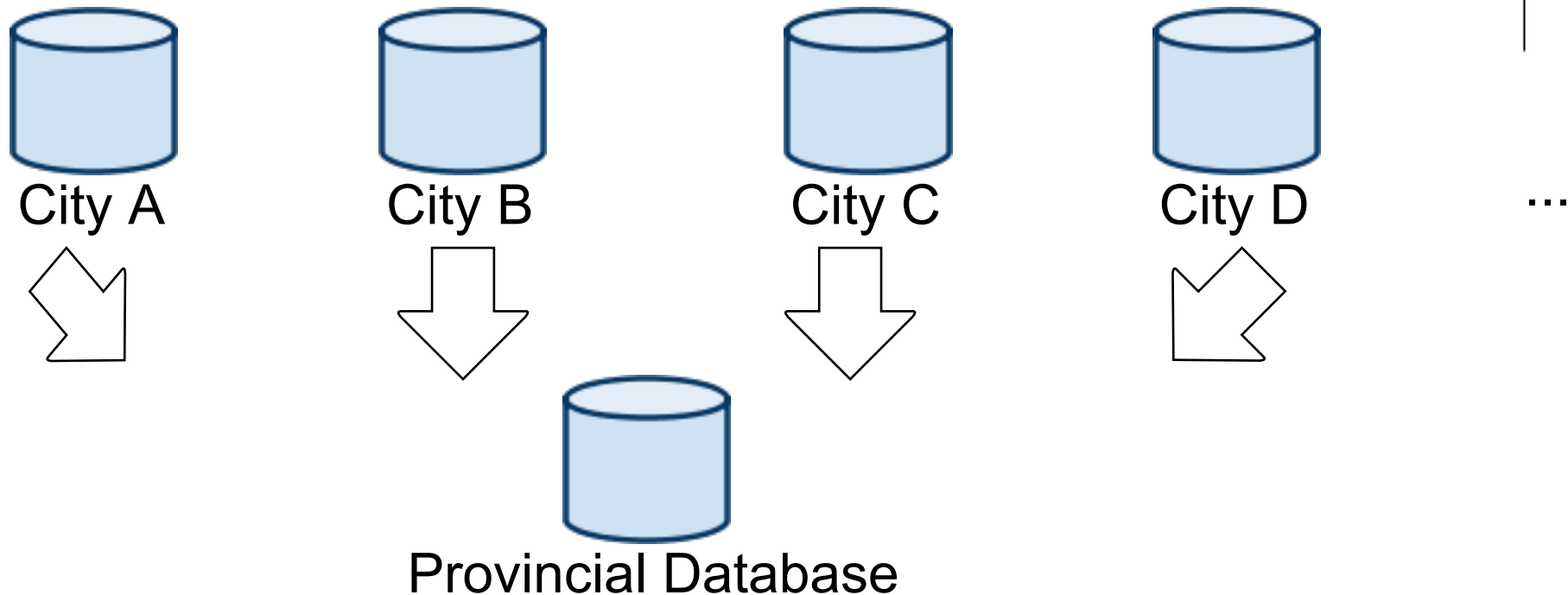


One config file:

```
[MAIN]
NAME=mydns
DSN0=dbname=mydns user=mydns password=... host=master
DSN1=dbname=mydns user=mydns password=... host=slave
```

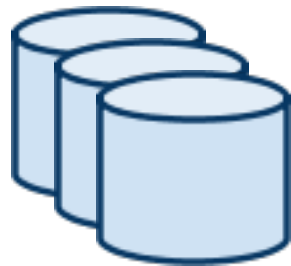
Projeto: 911

BD Consolidado

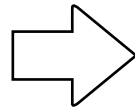


- Async - Cada cidade e autônoma - Backup
- Algumas modificação de dados (adicionar cidade de referência, ignorar exclusões, etc)
- Central de Estatísticas, Mapa do Crime (SIG)

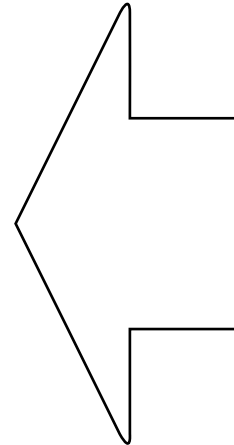
Caso de replicação simples para servidor OLAP Server em SIU



Masters
(Several
apps)



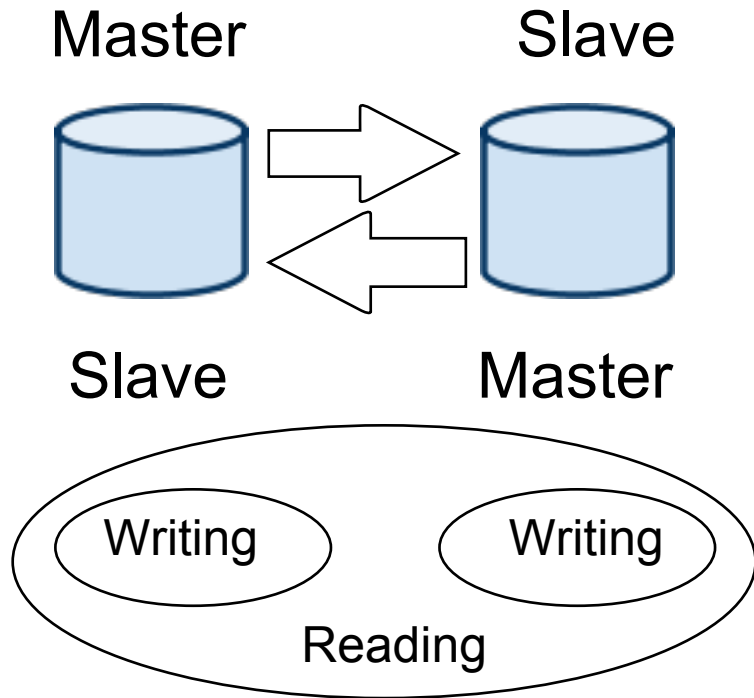
OLAP



Cube
Generation
/
Querys

- Várias aplicações replicam informações para um servidor OLAP em uma central, a geração do cubo será desenvolvida.

Estudo de caso: Multimaster



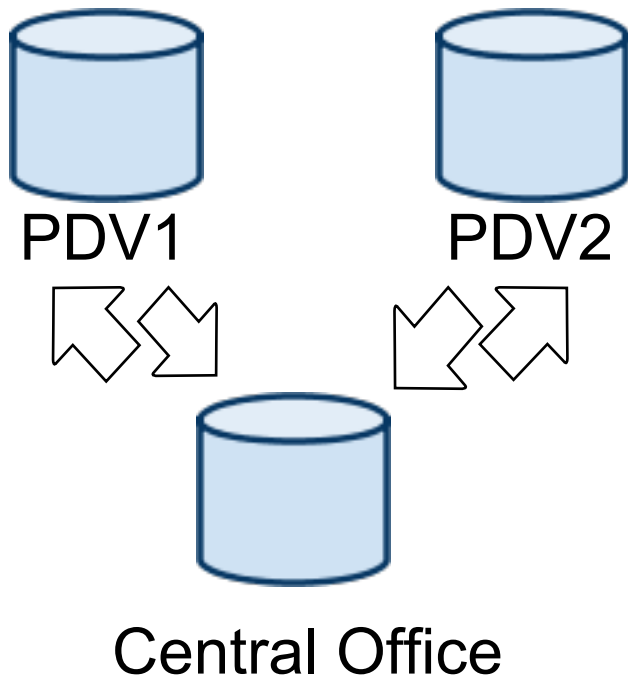
- Assíncrono
- Ambos master e slave
- 5GB
- tamanho total de cada BD
- 100K logs/dia (picos)
- Sequences: != starts

Two config files:

```
NAME=node1
DSN0=dbname=test user=replica pass... host=node1
DSN1=dbname=test user=replica pass... host=node2
SKIP_USER=replica # do not replicate replicated data :)
```

```
NAME=node2
DSN0=dbname=test ... host=node2
DSN1=dbname=test ... host=node1
SKIP_USER=replica
```

Estudo de caso: Repl. condicional

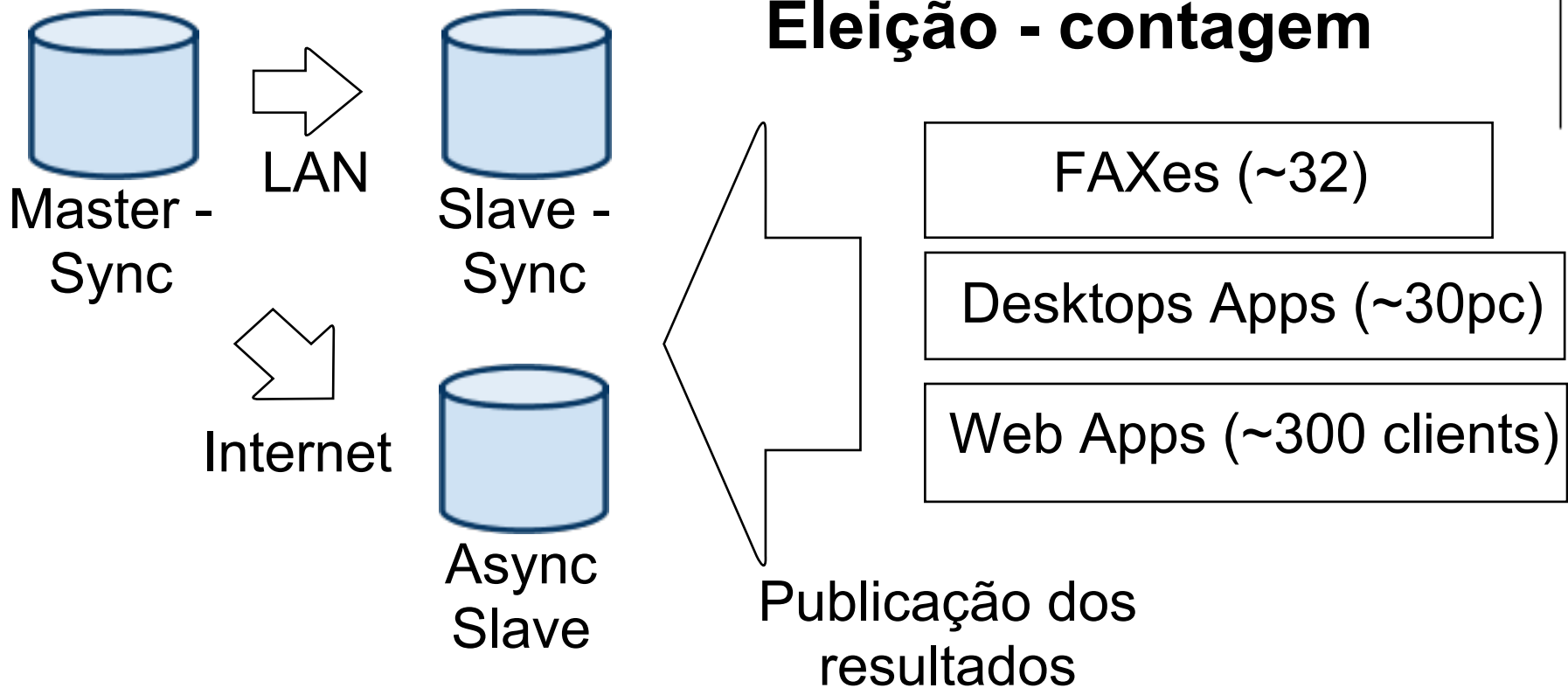


- Múltiplos pontos de vendas
- Assíncrono - Multimaster
- Cada PDV master and slave
- Único cliente de sincronização relacionados, ordens e suas atualizações
- 2 arquivos de configuração por PDV

replica_log_filter (extract from "central office"):

id	relname	event	condition	slave1	slave2	
1	order	INSERT	new['pos_id']==1	true	false	insert order in slave1 if new.pos_id=1
2	order	INSERT	new['pos_id']==2	false	true	insert order in slave2 if new.pos_id=2
3	order	UPDATE	old['pos_id']==1	true	false	update order in slave1 if old.pos_id=1
4	customer	UPDATE	True	true	true	update customer in slave1 and slave2
5	customer	DELETE	True	true	true	delete customer on slave1 and slave2

Estudo de caso: Sync + Async



- Sync: ALERCE (python apps), Async: PyReplica
- ~6 hs, 3k faxes, 3.5GB total size
- ~10kbps peak async replication (including BYTEA)
- Balanceamento de carga
- (R.O. apps directed to slaves)
- HA: Failover & Failback (plano de contingência)

¡Obrigado!



Contato:

- reingart@gmail.com
- efranco@siu.edu.ar

Links úteis:

- PyReplica: <http://pgfoundry.org/projects/pyreplica/>
- ArPUG: PostgreSQL Argentina: www.arpug.com.ar
- PyAr: Python Argentina: www.python.org.ar