Новые технологии репликации данных в PostgreSQL

Александр Алексеев
@afiskon

Два слова о себе

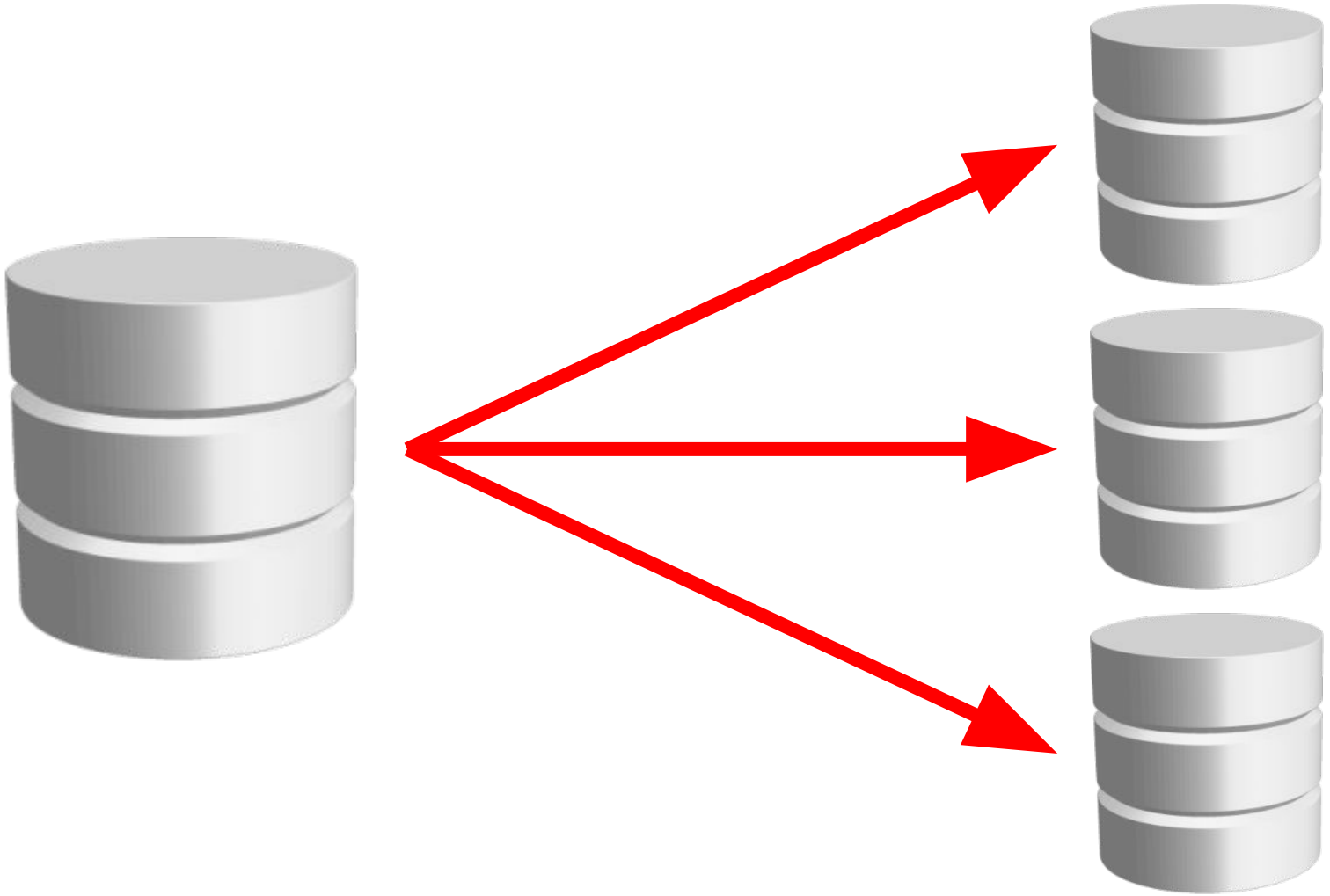
DevZen
PODCAST

Для кого этот доклад

- Вы считаете, что репликация настраивается непостижимо сложно.
- Вы никогда не настраивали репликацию в PostgreSQL или хотели бы освежить знания.
- Вы хотели бы знать, что тут появилось нового за последнее время.

Бонусы!

Репликация



Зачем это нужно

- Распределение нагрузки
  - OLTP: на чтение ходим в реплики
  - OLAP: тяжелая аналитика на реплике
  - Снятие бэкапов
- Фейловер / HA
  - Ручной (лучше!) или автоматический
- Отложенная репликация
- **Не заменяет резервное копирование!**

Коротко о главном

- **Потоковая репликация**
  - **Асинхронная**
    - Используется по умолчанию. Быстро, но можно потерять данные.
  - **Синхронная**
    - Медленнее (в рамках одного ДЦ не намного), но надежнее. Желательно иметь две реплики.
  - **Каскадная**

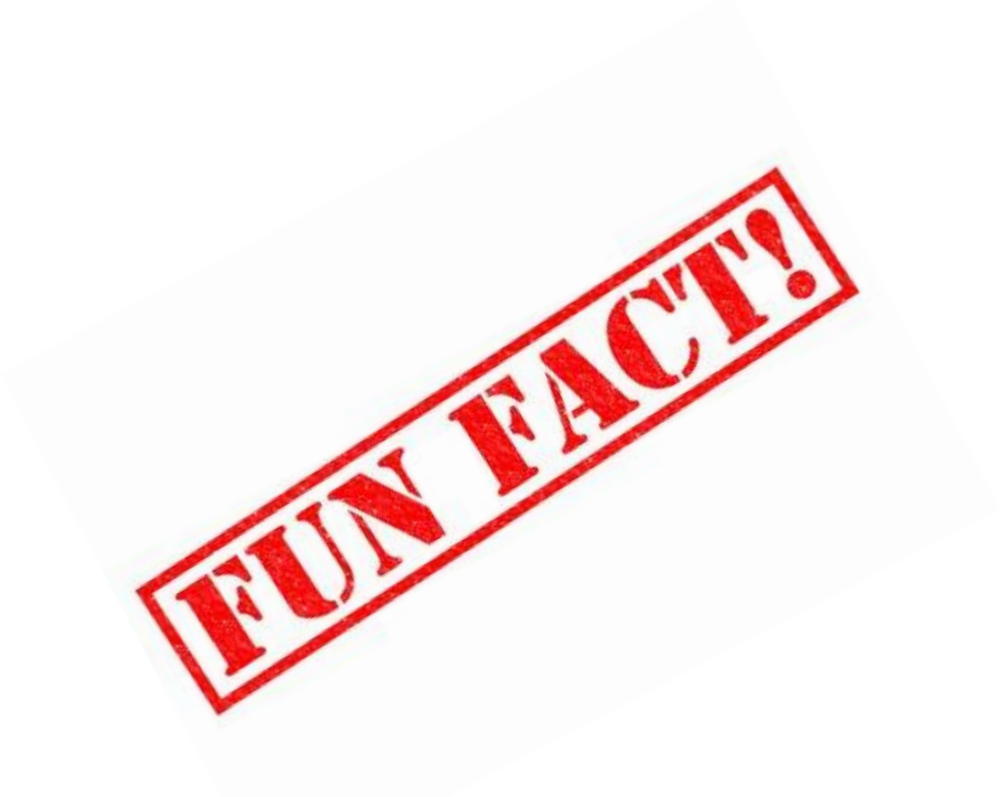
Коротко о главном

- **Логическая репликация**
  - **Slony**
    - Медленно, стабильно.
  - **pglogical**
    - Быстро, но пока что сыро.

Fun fact!

Потоковая репликация:

- Не работает между разными архитектурами
- Не работает между разными версиями PostgreSQL



Настройка master - pg_hba.conf

```
host  replication postgres  10.0.3.0/24    md5

host  all         postgres  10.0.3.0/24    md5
```

Настройка master - postgresql.conf

```
wal_level = hot_standby

# это нужно, чтобы работал pg_rewind

wal_log_hints = on

max_wal_senders = 8 # default: 0

# если хотим синхронную репликацию

# на одну любую реплику

synchronous_standby_names = '*'

hot_standby = on
```

9.6: synchronous_standby_names

```
synchronous_standby_names = '2(alpha, beta, gamma)'
```

(например, можно сделать репликацию на кворум)

Настройка slave - pg_basebackup

```
cd /var/lib/postgresql/9.5/

rm -rf main

mkdir main

chmod go-rwx main

pg_basebackup -P -R -X stream -c fast \
  -h 10.0.3.245 -U postgres -D ./main
```

(не поленитесь почитать ман, он классный!)

Настройка slave - main/recovery.conf

```
# по умолчанию восстанавливаемся до текущего
# таймлайна, а нам нужно до максимального
# таймлайна в архиве
recovery_target_timeline = 'latest'
# чтобы отставать от мастера
recovery_min_apply_delay = 10min
```

(+ правим pg_hba.conf и postgresql.conf аналогично мастеру)

Проверка репликации

На мастере:

```
SELECT * FROM pg_stat_replication;
```

На слейве:

```
SELECT now()-pg_last_xact_replay_timestamp();
```

Где я (false - на мастере, true - на реплике):

```
SELECT pg_is_in_recovery();
```

Полезные параметры

Не дожидаемся fsync на реплике:

```
SET synchronous_commit = 'remote_write';
```

Пишем только локально, не ждем подтверждения от реплики:

```
SET synchronous_commit = 'local';
```

Возвращаем обычное поведение:

```
SET synchronous_commit = 'on';
```

9.6: дожидаемся, пока изменения применятся к таблицам:

```
SET synchronous_commit = 'remote_apply';
```

Промоутим реплику до мастера

```
pg_ctl promote -D /var/lib/postgresql/9.5/main/
```

… и смотрим в логи.

Все равно лучше сделать restart, чтобы клиенты перепроверили, куда они подключены - к мастеру или слейву.

Переключение реплик на новый мастер

```
pg_rewind \
  -D /var/lib/postgresql/9.5/main/ \
  --source-server="host=?? port=?? user=?? password=??"


cd /var/lib/postgresql/9.5/main/
mv recovery.done reconvery.conf
vim recovery.conf
# проверяем IP мастера и наличие строчки:
# recovery_target_timeline = 'latest'
```

ERROR

Если видим:

```
requested WAL segment 00200000005 has already been removed
```

Значит, реплика слишком сильно отстала. Вместо pg_rewind нужно использовать pg_basebackup, как было описано выше.

Автофейловер

На что стоит посмотреть:

- http://www.repmgr.org/
- https://github.com/sorintlab/stolon

Вопросы?

Что почитать (бонусный слайд)

Про PostgreSQL:

- **Официальная документация!**
- **PostgreSQL 9.0 High Performance**, Gregory Smith
- **PostgreSQL Server Programming**, Hannu Krosing et al

Про РСУБД:

- **Database System Implementation**, Hector Garcia-Molina et al

Про распределенные системы:

- **Distributed systems for fun and profit**, Mikito Takada
- **Introduction to Reliable and Secure Distributed Programming**, Christian Cachin et al

Про NoSQL-решения (бонусный слайд)

Почему неплохо бы держать рядом NoSQL:

- **Автошардинг / решардинг (Amazon RDS? Heroku?)**
- **Счетчики, сессии и коды капч не стоит держать в РСУБД (блоатинг,** `rand() % 1000 == 0`**)**
- **Колоночные хранилища и транзакции несовместимы**
- **Для неизменяемых данных не очень-то нужны транзакции**
- Полнотекстовый поиск - ОК (GIN, GiST индексы)
- Очереди - ОК (SELECT FROM … FOR UPDATE SKIP LOCKED)

Что посмотреть (не в качестве основной СУБД):

- [+] Memcached, Redis, RethinkDB (драйверы?), CockroachDB (?)
- [+/-] Tarantool (мало использовал), Cassandra (200 мс stop the world?)
- [--] Couchbase, MongoDB, Riak

См также: Sphinx, ElasticSearch, RabbitMQ, Storm / Heron

Колоночные хранилища? Таймсерии?