Postgresql и отказоустойчивость

Кулагин Михаил

Postgres Pro

postgrespro.ru

Обо мне

Кулагин Михаил, DBA, Postgres Professional

Занимаюсь внедрением отказоустойчивых решений на основе PostgreSQL

О чём?

Расскажу о:

- различных способах обеспечения отказоустойчивости СУБД PostgreSQL
- плюсах и минусах этих способов
- подробнее расскажу отказоустойчивом стеке pacemaker/corosync и multimaster

Отказоустойчивость

Postgres PROFESSIONAL

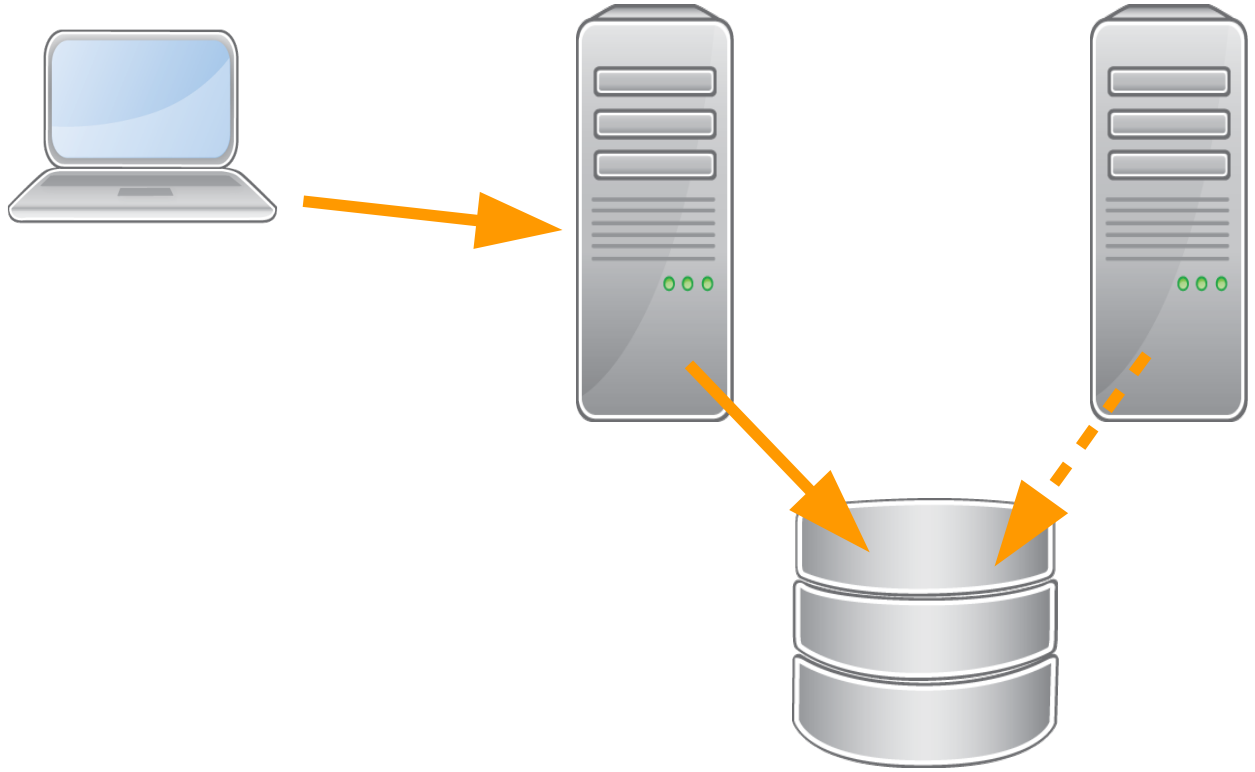
Отказоустойчивость — свойство технической системы сохранять свою работоспособность после отказа одного или нескольких составных компонентов. Отказоустойчивость определяется количеством любых последовательных единичных отказов компонентов, после которого сохраняется работоспособность системы в целом.

Базовый уровень отказоустойчивости подразумевает защиту от отказа одного любого элемента — исключение единой точки отказа. Основной способ повышения отказоустойчивости — избыточность.

Подходы

- Общий диск
- Репликация файловой системы
- Репликация на основе триггеров
- Распределение запросов
- Журналирование транзакций
- Логическая репликация
- Мультимастер

Общий диск



Сделаем диск доступным с нескольких серверов и, при сбое, воспользуемся аварийным восстановление СУБД

Write-ahead log (WAL)

```
$ pg_xlogdump 000000010000000000000004 ...
rmgr: Heap len (rec/tot): 3/ 59, tx: 1838, lsn: 0/0400FD78, prev
0/0400FCD8, desc: INSERT off 4, blkref #0: rel 1663/16384/16386 blk
0
rmgr: Btree len (rec/tot): 2/ 64, tx: 1838, lsn: 0/0400FDB8, prev
0/0400FD78, desc: INSERT_LEAF off 4, blkref #0: rel
1663/16384/16392 blk 1
rmgr: Transaction len (rec/tot): 8/ 34, tx: 1838, lsn: 0/0400FDF8, prev
0/0400FDB8, desc: COMMIT 2016-04-06 12:43:07.672658 UTC
```

Общий диск

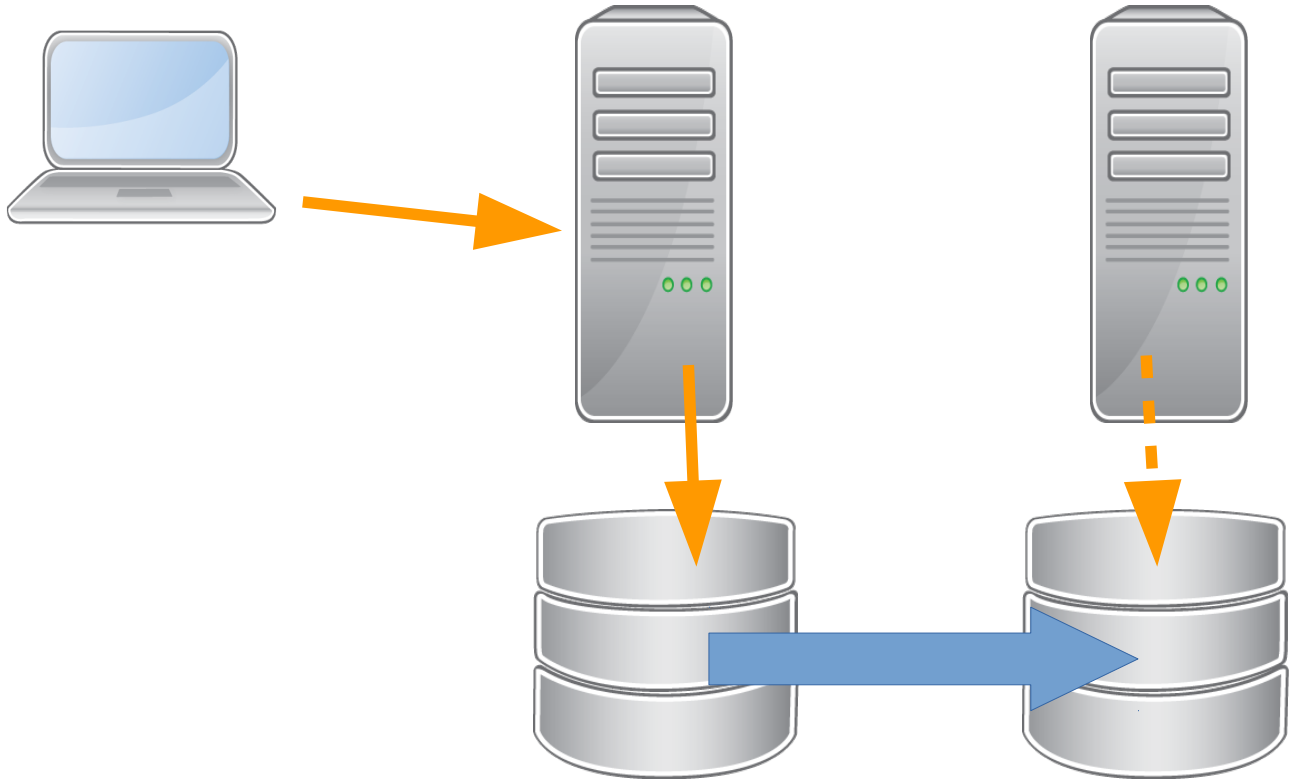
Плюсы:

- нет конфликтов записи
- не нужна синхронизация между серверами

Минусы:

- одна точка отказа
- резервный сервер простаивает
- failover относительно долгий

Репликация файловой системы



У каждого сервера будет своя копия данных, репликация средствами файловой системы или блочного устройства (например, DRBD)

Репликация файловой системы

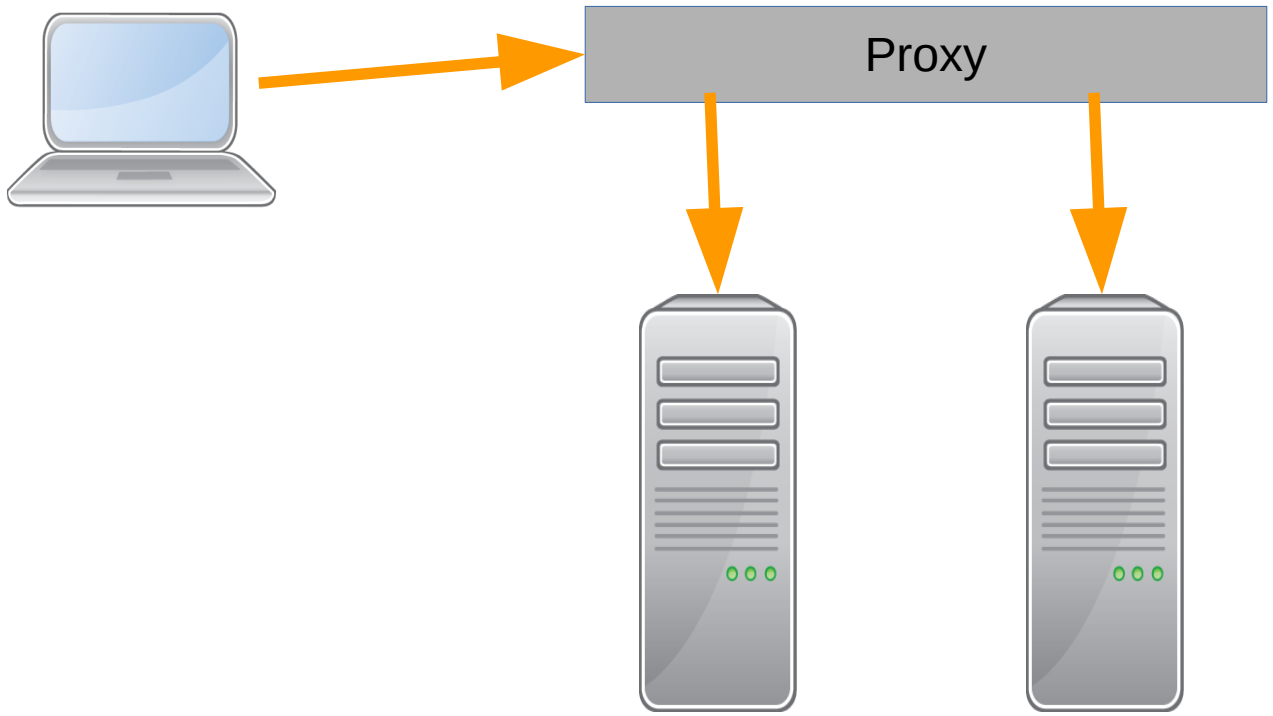
Плюсы:

- дополнительного оборудования не требуется
- без потери данных

Минусы:

- резервный сервер простаивает
- временные потери на синхронизацию файловой системы/блочного устройства
- failover относительно долгий

Распределение запросов



Поставим proxy-сервер, который будет дублировать запросы на запись на все сервера (например, PgPool-II)

Распределение запросов

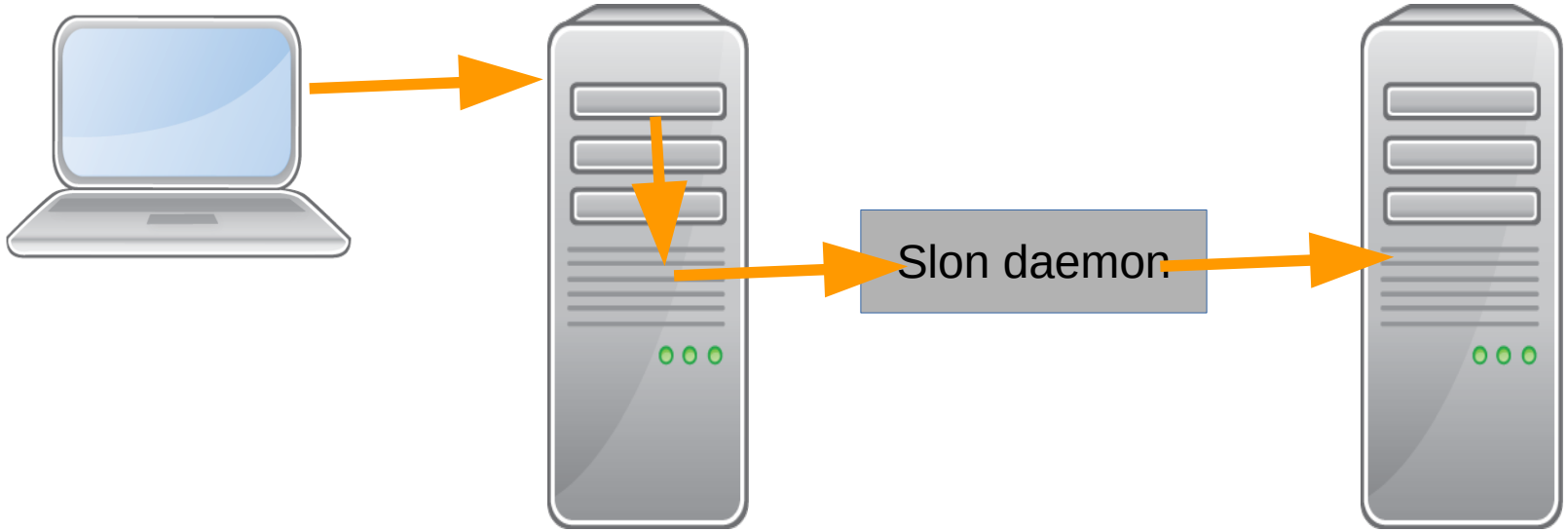
Плюсы:

- без потери данных
- резервный сервер может обслуживать запросы на чтение

Минусы:

- возможно неоднозначное исполнение запросов на серверах
- возможны расхождение реплик по данным и конфликты

Trigger based replication



Залогировать изменения в данных, используя триггеры БД, и передать их на реплику (Slony-I)

Trigger based replication

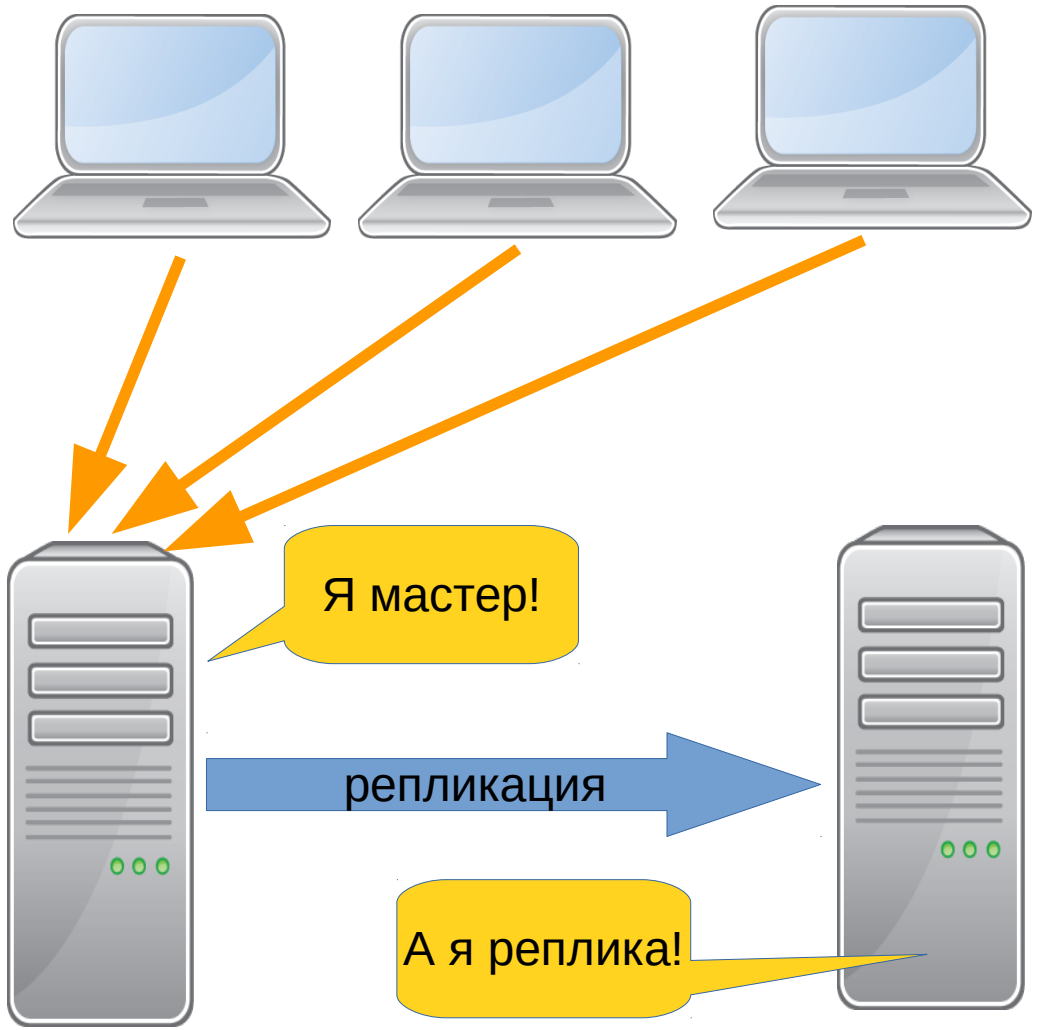
Плюсы:

- дополнительного оборудования не требуется
- резервный сервер может обслуживать запросы на чтение
- можно избирательно реплицировать данные
- позволяет смешивать разные версии СУБД

Минусы:

- дополнительная нагрузка на мастер (существенная)
- возможна потеря данных

Streaming replication



Потоковая репликация

Два режима работы реплики:

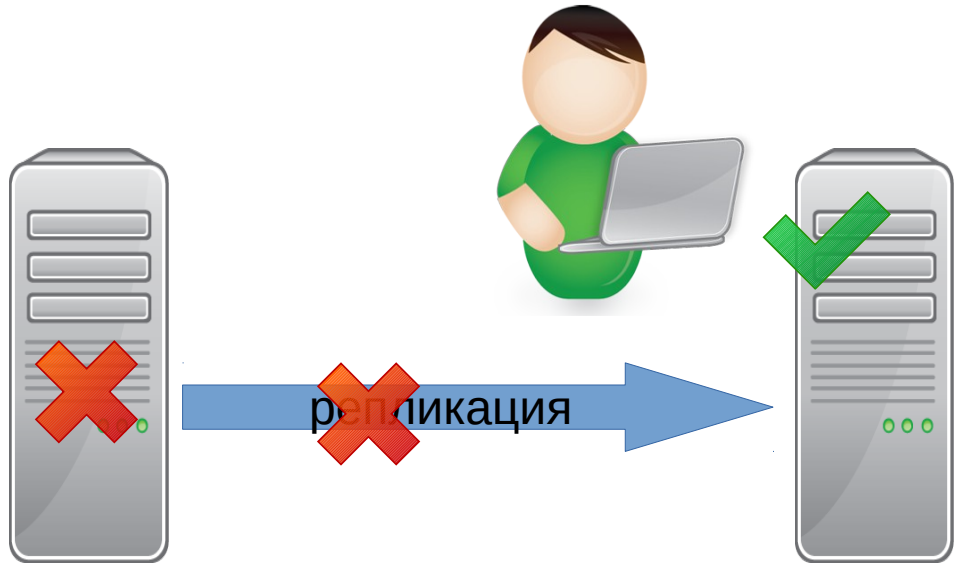
- Асинхронная
- Синхронная

Управление потоковой репликацией

Всё хорошо?

При сбое мастера необходимо вручную:

1. обнаружить сбой
2. найти реплику, которая была синхронной
3. выполнить promote
4. переконфигурировать остальные асинхронные реплики на новый мастер



Streaming replication

Плюсы:

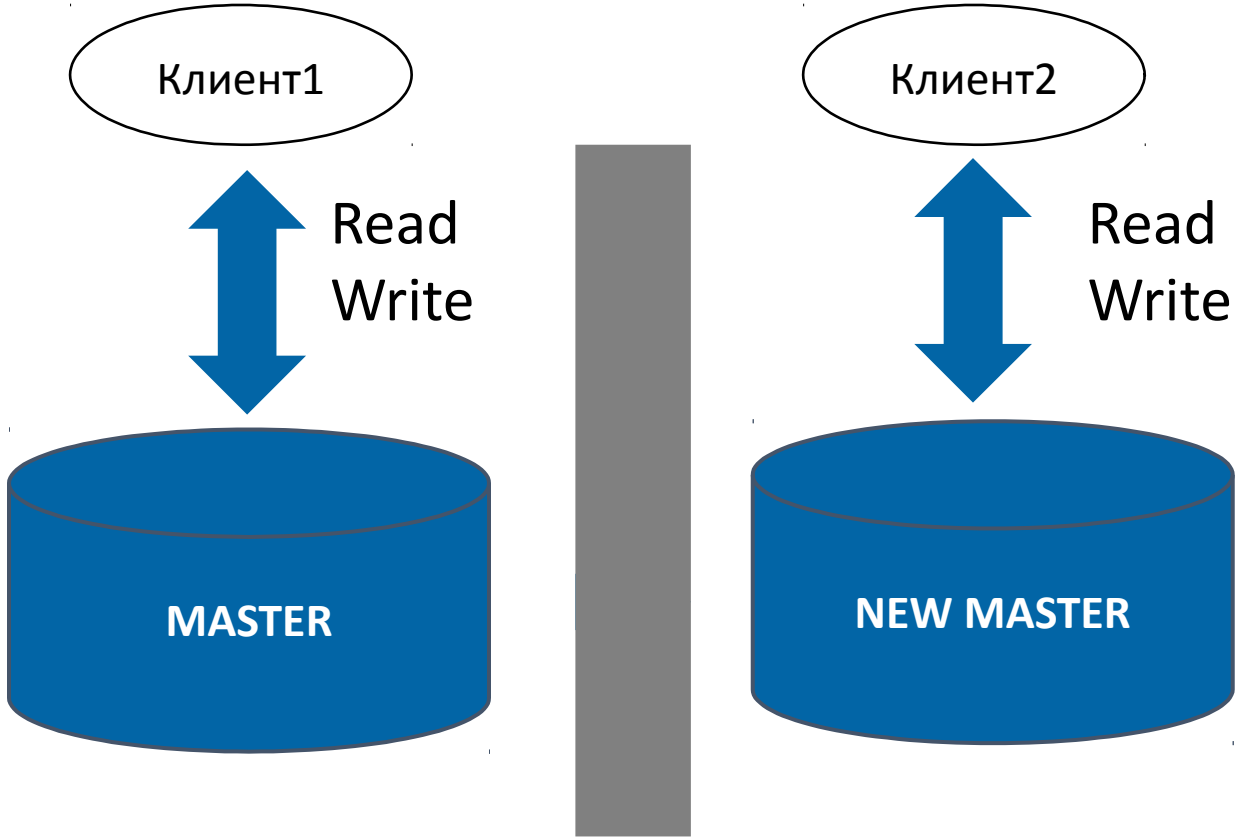
- штатная возможность в PostgreSQL
- дополнительного оборудования не требуется
- без дополнительной нагрузки на мастер
- резервный сервер может обслуживать запросы на чтение
- можно выбрать синхронный вариант репликации (без потери данных, но с задержкой при подтверждении commit'a), асинхронный (без задержки, но с возможной потерей данных) или комбинацию синхронных и асинхронных реплик

Минус:

- Необходимо вручную перенастраивать реплики

Основная боль отказоустойчивых кластеров

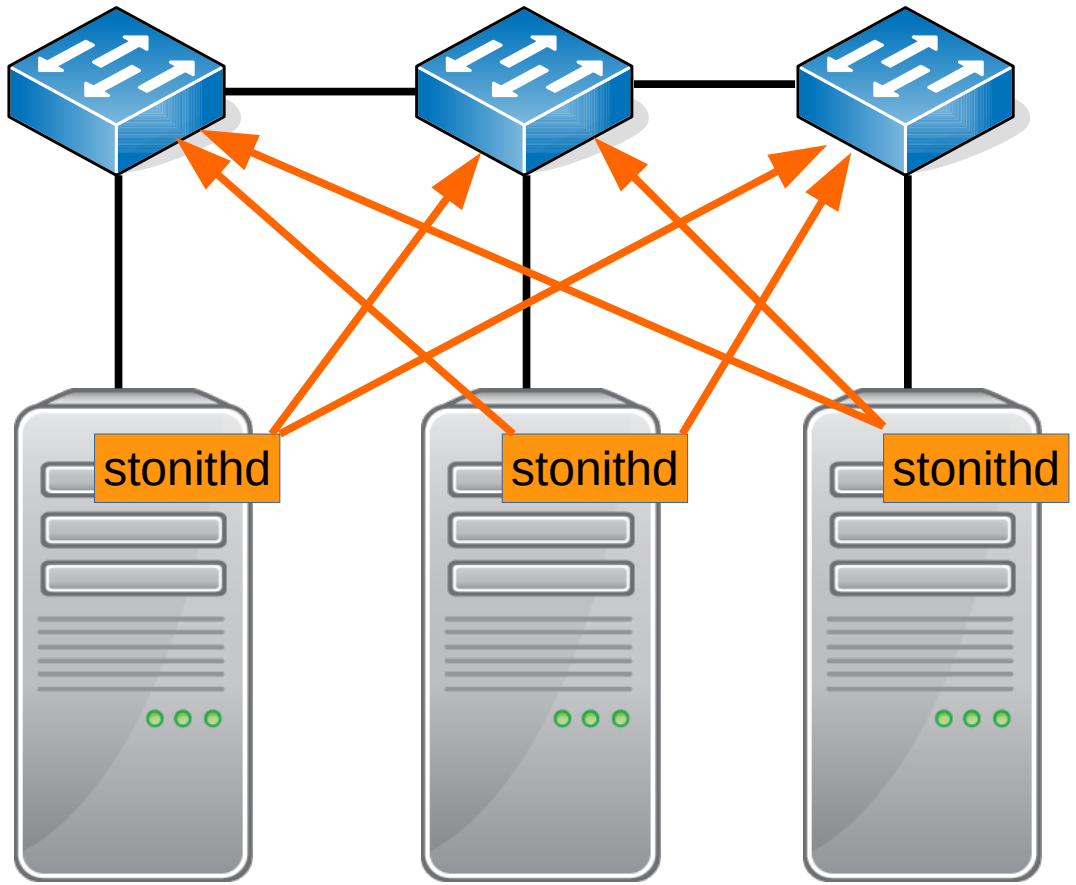
- Split-brain (разрыв мозга) :



Временное или длительное прерывание внутренней связи между узлами:

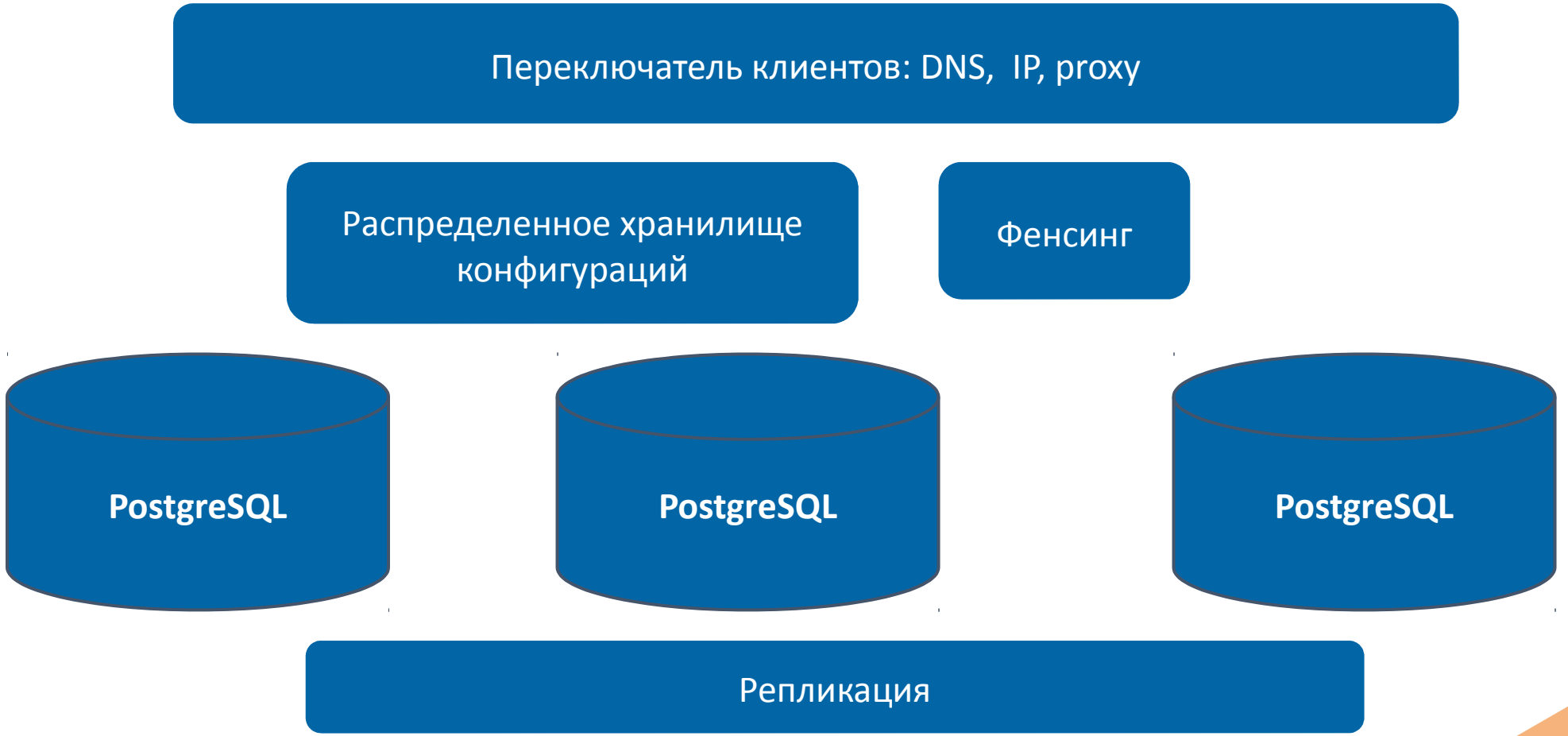
- Синхронная реплика становится мастером
- Часть клиентов работает со старым мастером
- Другие клиенты соединяются с новым
- Хаос нарастает

Fencing

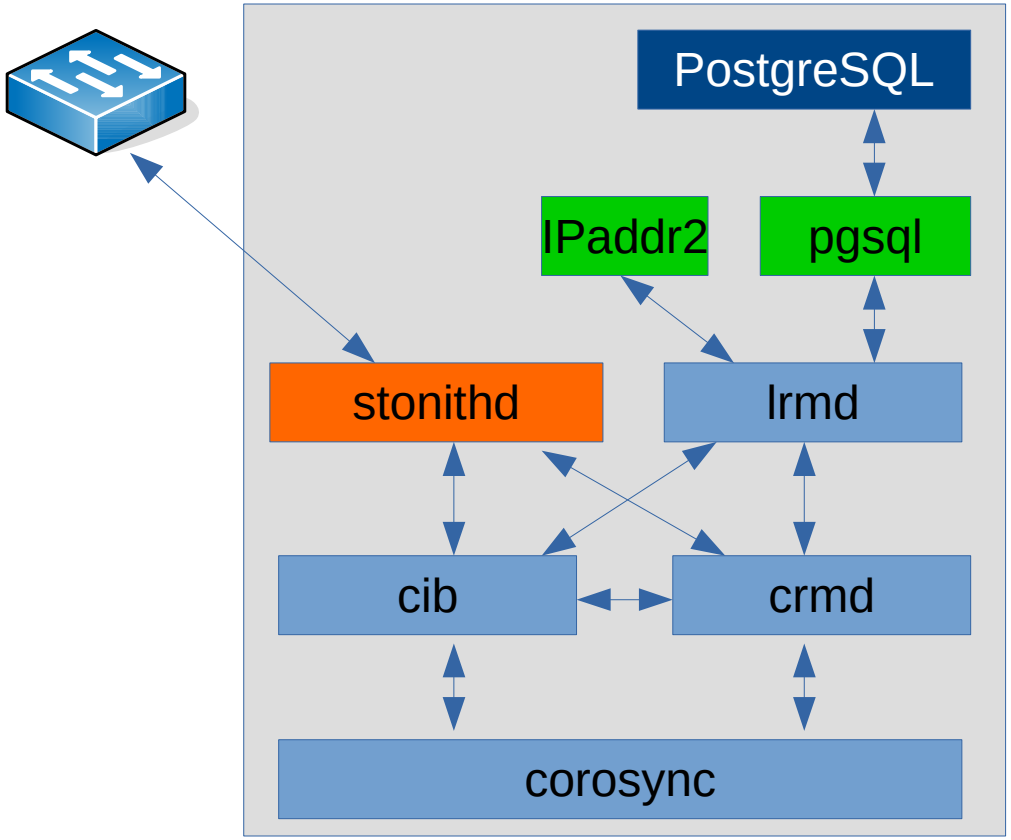


Fencing – способ отключить сбойный узел внешним для этого узла способом.

Компоненты надежного кластера



Стек pacemaker/corosync



Logical replication

Аналогично триггерной, но в качестве источника использовать logical decoding (UDR, BDR, pglogical)

Logical

```
$ pg_recvlogical --create-slot --slot test -d test
test=> insert into a values(2);
INSERT 0 1
$ pg_recvlogical --slot test -d test --start -f - BEGIN 83668643
table public.a: INSERT: b[integer]:2 COMMIT 83668643
^C
$ pg_recvlogical --drop-slot --slot test -d test
```

Logical replication (UDR/BDR)

Плюсы:

- Нет накладных расходов триггерной репликации
- дополнительного оборудования не требуется
- резервный сервер может обслуживать запросы на чтение
- можно избирательно реплицировать данные
- позволяет смешивать разные версии СУБД

Минусы:

- Возможна потеря данных (нет механизма для блокировки изменения одних и тех же записей, есть только стратегия разрешения конфликтов)

Postgres Pro Мультимастер

- Основан на логической репликации
- Каждая транзакция реплицируется на каждый узел
- Распределенная транзакция
- Встроенный failover
- Расширение Postgres Pro
- Коммерческая лицензия, часть – Open source
- Транзакционная целостность ЕСТЬ
- Масштабируемость по чтению: отличная
- Масштабируемость по записи: планируется

Часть Postgres Pro Enterprise (коммерческий продукт)

Multimaster внутри

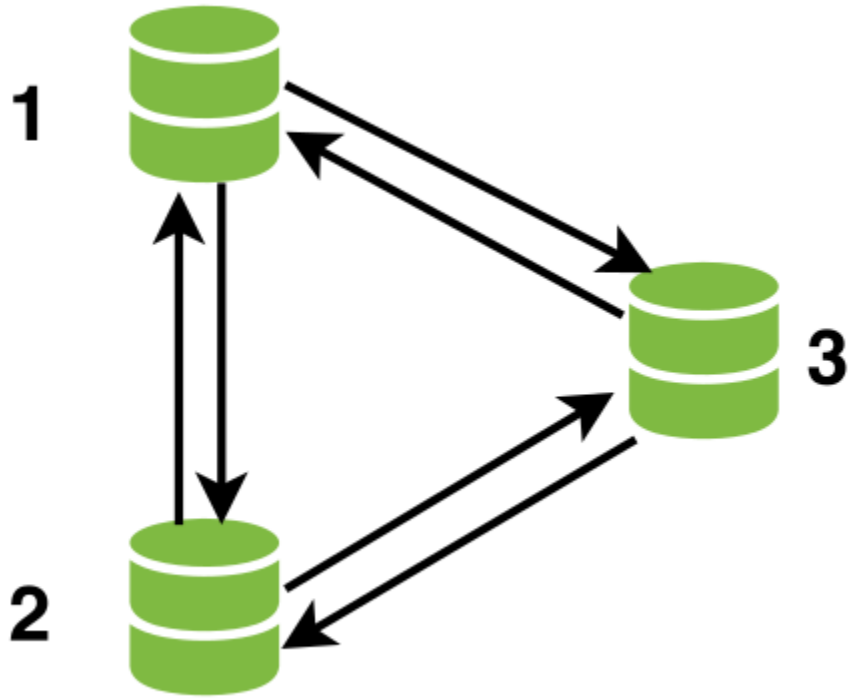
Репликационные процессы:

- wal sender process (на каждом узле N-1)
- mtm_pglogical_receiver (N-1)
- bgw_pool_worker (max_connection*(N-1))

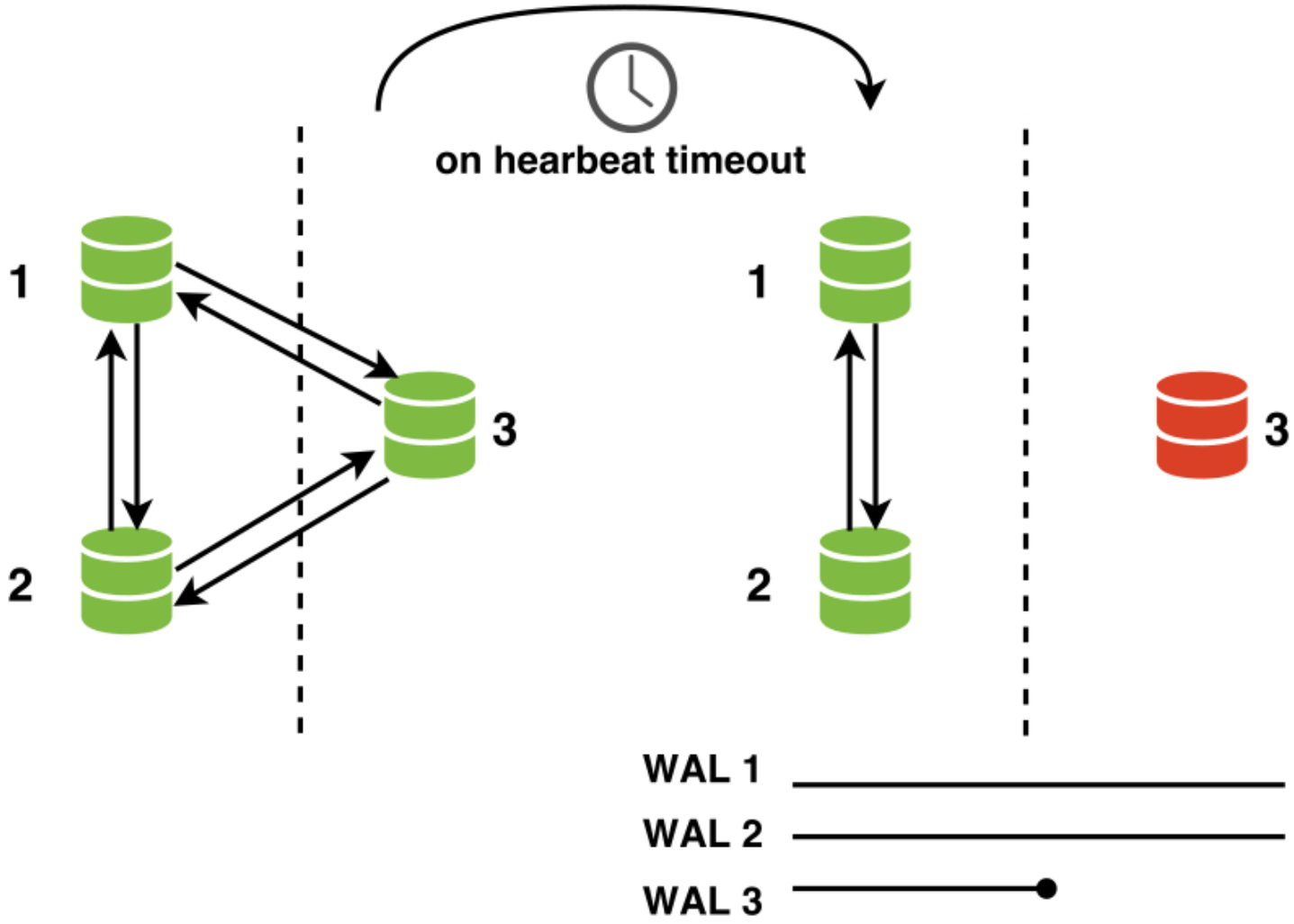
Процессы арбитра (по одному на каждый узел):

- Mtm-monitor
- Mtm-receiver
- Mtm-sender

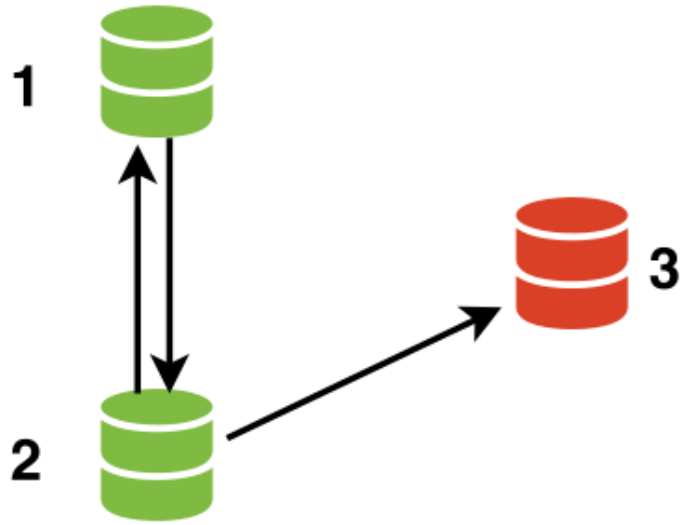
Нормальная работа

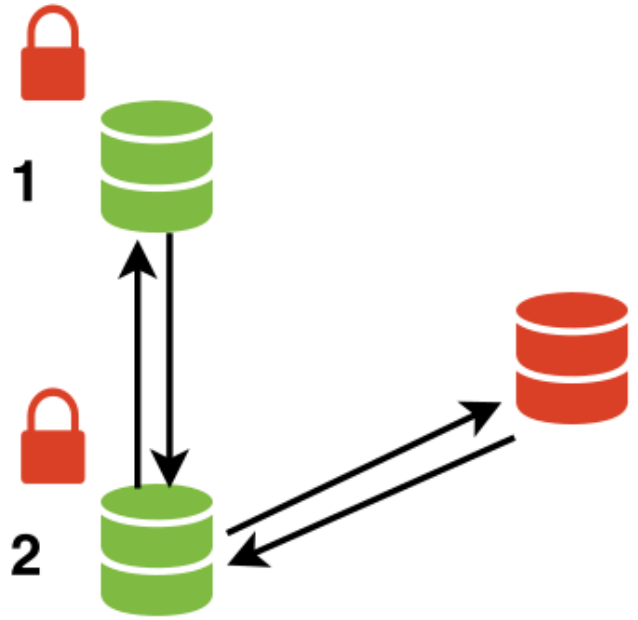


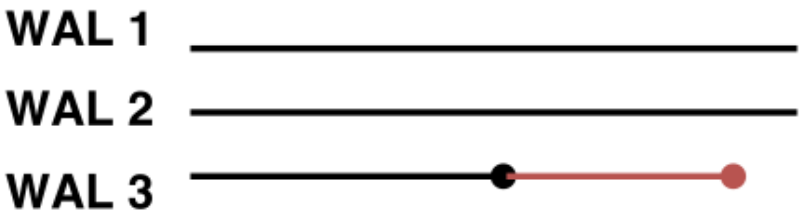
Разрыв внутренней сети

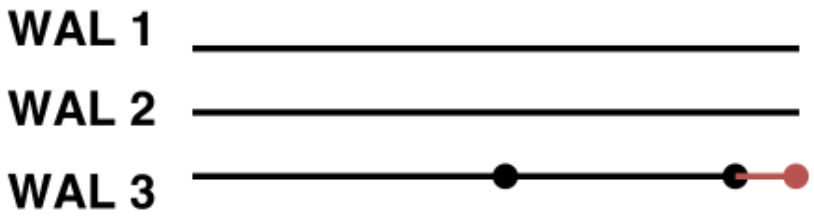


Восстановление после разрыва сети

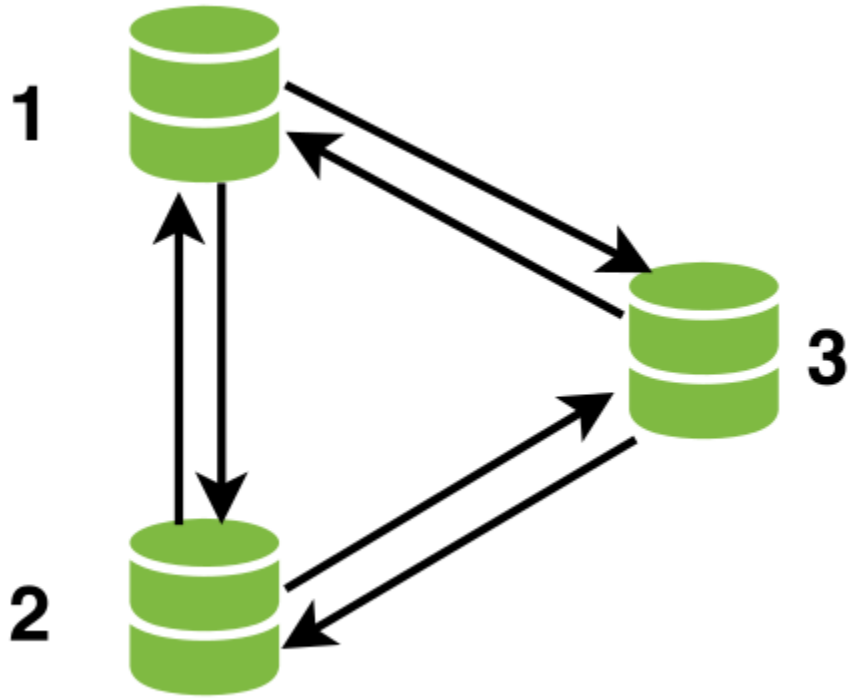








Нормальная работа после восстановления



Мультмастер Postgres Pro – самый простой в использовании кластер

- Нет накладных расходов на чтение.
- Транзакция может исполняться на любом узле, и будет реплицирована на все узлы.
- Автоматический failover (требуется реконнект клиента)

Особенности мультимастера Postgres Pro

- Одинаковые данные на всех узлах (пока)
- Возможность иметь локальные таблицы на узле
- Максимальная совместимость с постгресом
- DML и DDL на любом узле

- Следующий шаг: интеграция шардинга для масштабируемости по записи

Подтвержденные фичи

- Запуск-остановка узла
- Восстановление внезапно выключенного узла
- Простой разрыв сети
- Асимметричный разрыв сети
- Сдвиг времени
- Различие скорости хода времени на узлах (в работе)

Производительность

- Чтение – так же, как на независимых узлах
- Коммит требует времени (+2 раундтрипа по сети).
- Большие транзакции тормозят (но это пройдет)

Ограничения multimaster

- Уровни изоляции: READ COMMITTED, REPEATABLE READ
- Блокировка может «всплыть» при коммите.
- Последовательности «по модулю N»
- DDL реплицируется чуть иначе чем DML

Отличие поведения

```
BEGIN;
CREATE TABLE q (z int primary key);
COMMIT;
---- тут висим
---- проскочило
```

```
BEGIN;
CREATE TABLE q (z int primary key);


COMMIT;
ERROR:  [MTM] Transaction MTM-3-348-2
(1695) is aborted by DTM
```

Полезные ссылки

https://postgrespro.ru/docs

https://postgrespro.ru/education/courses

https://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling

http://clusterlabs.org/

https://www.2ndquadrant.com/en/resources/pglogical/

https://postgrespro.ru/docs/postgresproee/9.6/multimaster.html

Вопросы?

Postgres Professional

http://postgrespro.ru/

+7(495)1500691

info@postgrespro.ru

postgrespro.ru