



Репликация: варианты



Подключение нескольких реплик

Синхронная репликация

Каскадная репликация

Отложенная репликация

Несколько реплик

Задача

повысить доступность и распределить нагрузку

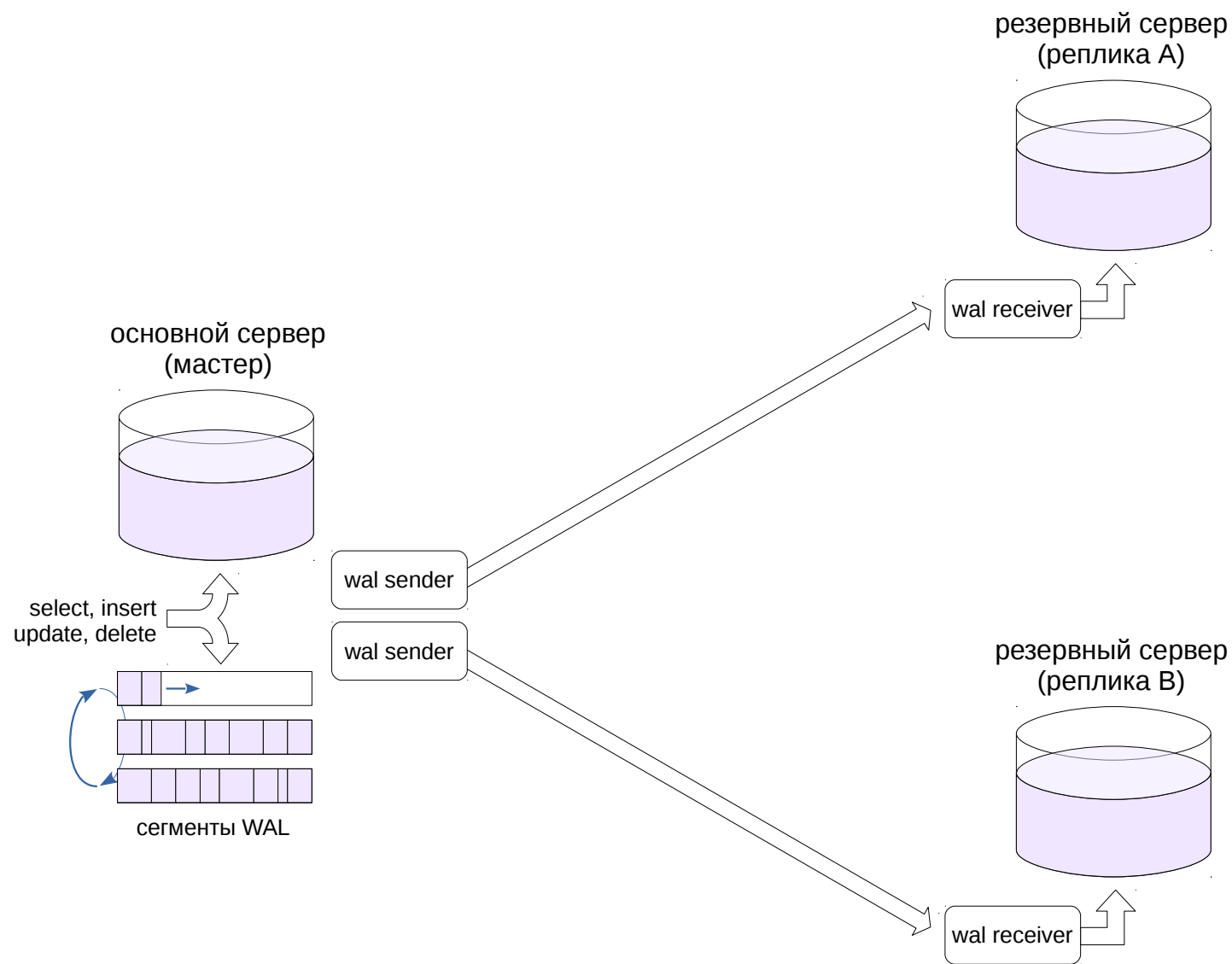
Механизм

несколько реплик подключаются к одному мастеру

Ограничения и особенности

на каждую реплику требуется отдельный процесс wal sender и отдельный слот репликации

Несколько реплик



Несколько реплик

Мастер

postgresql.conf

max_wal_senders

max_replication_slots

Реплика

дополнительные настройки не требуются

Задача

обеспечить надежность хранения данных

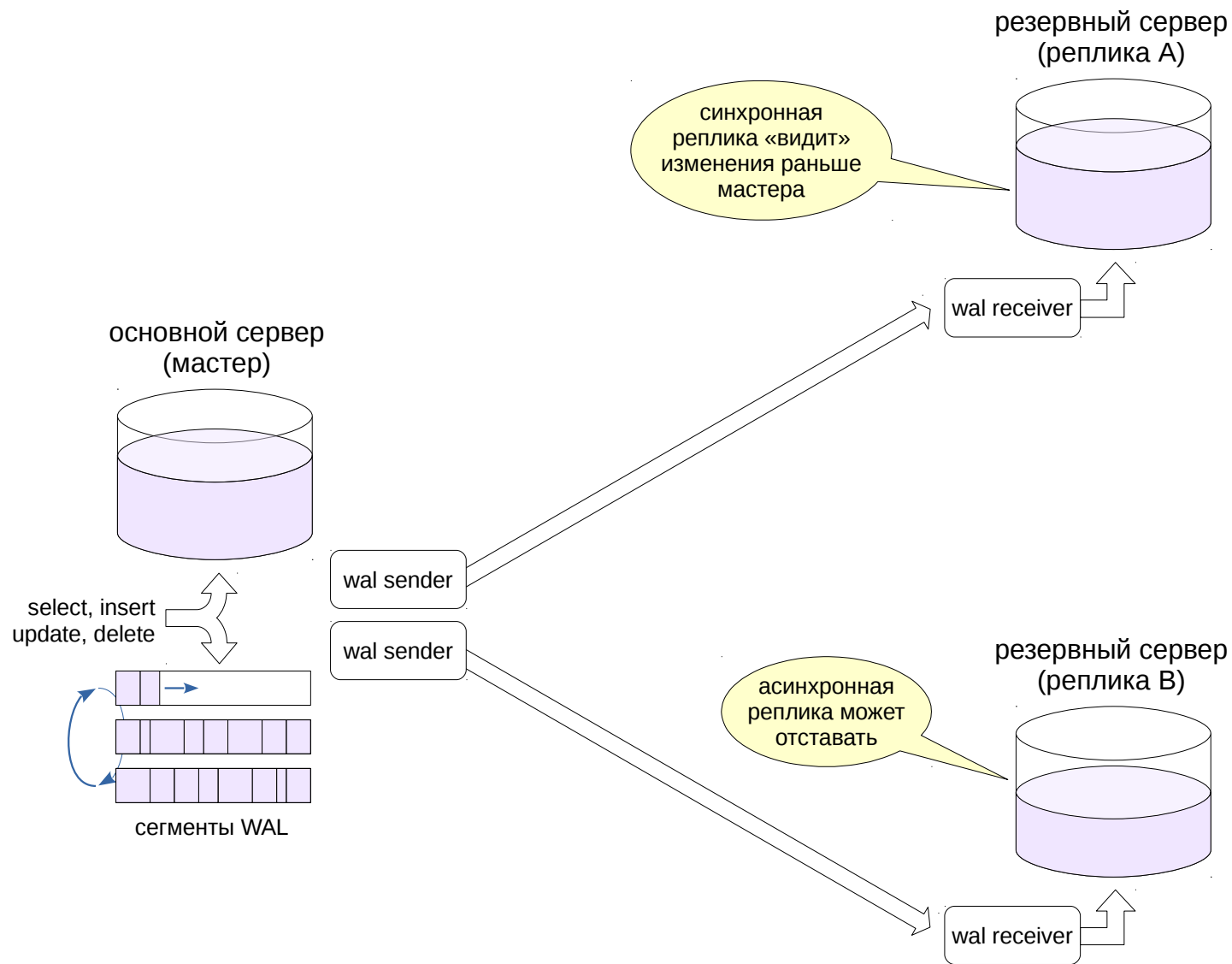
Механизм

при фиксации изменений мастер получает подтверждение от реплики, прежде чем вернуть управление

Ограничения

обеспечивается надежность, но не согласованность
синхронизация возможна только с одной репликой
при отключении одной синхронной реплики мастер начинает синхронизироваться со следующей из списка

Синхронная репликация



Мастер

postgresql.conf

synchronous_standby_names

список синхронных реплик (application_name) через запятую
звездочка (*) обозначает всех, но осторожнее с pg_receivexlog

synchronous_commit

off, local	выключено
remote_write	подтверждение записи в файл (в буфер ОС)
on	подтверждение попадания файла на диск

Реплика

recovery.conf

primary_conninfo = '... application_name=*имя* ...'

Особенности

фиксация занимает время, производительность мастера падает

— можно включать синхронизацию только для критичных транзакций

```
set local synchronous_commit = on;
```

— синхронную реплику лучше не нагружать

если ни одна реплика из списка синхронизации не подключена,

фиксация изменений никогда не завершится

— стоит иметь несколько реплик

синхронная реплика содержит наиболее полный набор данных

— именно на нее надо переключаться с мастера

Задача

иметь несколько реплик, не нагружая мастер

Механизм

одна реплика передает полученные данные другой реплике

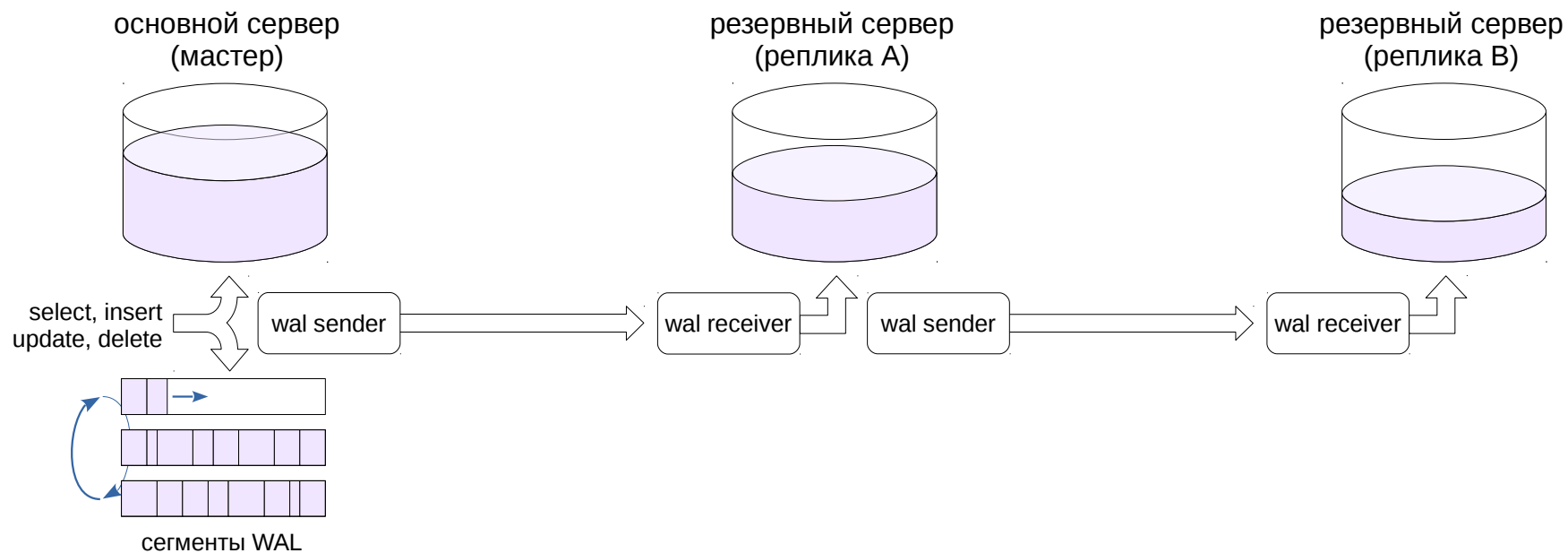
Ограничения и особенности

не поддерживается синхронная репликация

мастеру поступает обратная связь от всех реплик по цепочке

переключаться с мастера имеет смысл на ближайшую реплику

Каскадная репликация



Мастер

дополнительные настройки не требуются

Промежуточная реплика

postgresql.conf

max_wal_senders

max_replication_slots

pg_hba.conf

разрешение для базы replication

recovery.conf

recovery_target_timeline = 'latest'

Задача

«машина времени» и возможность восстановиться на определенный момент в прошлом без архива

Механизм

искусственная задержка применения записей WAL

Ограничения и особенности

нужна синхронизация часов между репликами
откладывается применение записей о фиксации (многоверсионность)
при окончании восстановления остаток применяется без задержки
включенная обратная связь при большом интервале приведет к увеличению размера таблиц на мастере из-за старых версий строк

Отложенная репликация



Отложенная репликация

Мастер

дополнительные настройки не требуются

Реплика

recovery.conf

`recovery_min_apply_delay`

а также функции:

`pg_xlog_replay_pause()`

`pg_xlog_replay_resume()`

Несколько реплик увеличивают отказоустойчивость и позволяют распределять нагрузку

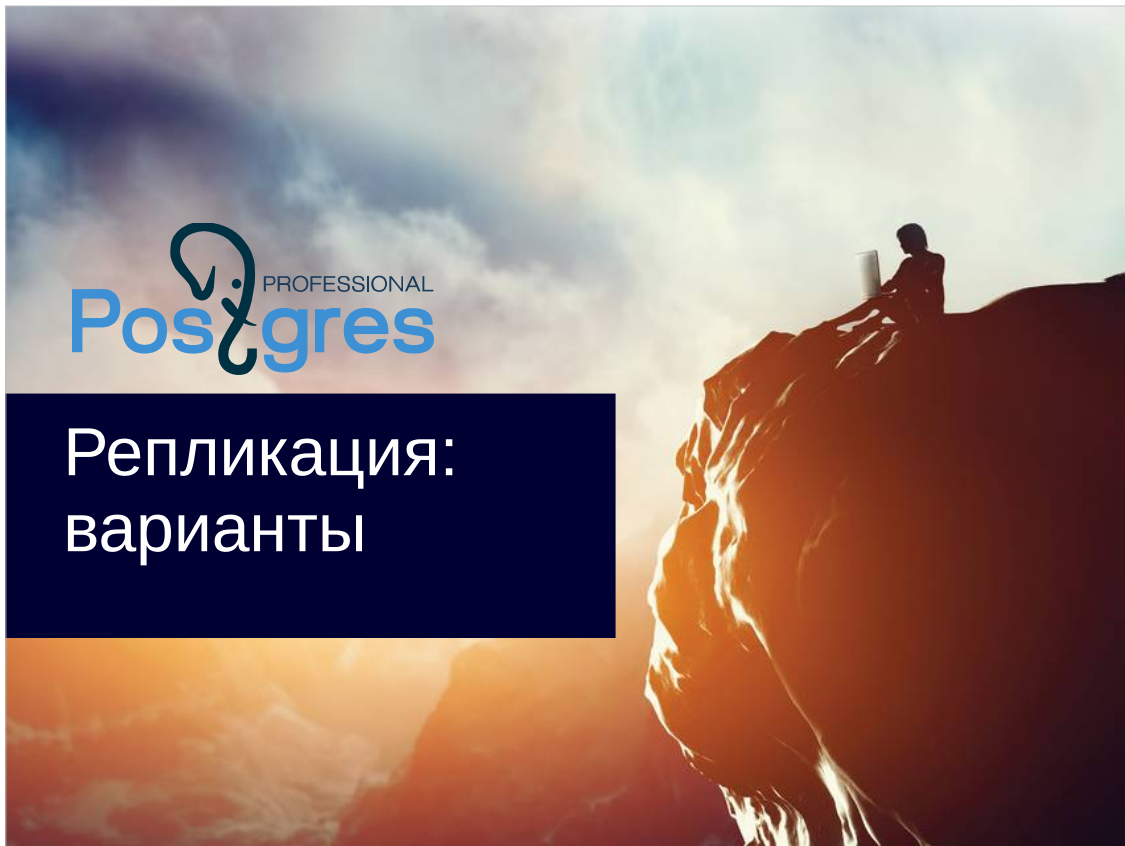
Синхронная репликация увеличивает надежность, но нужно учитывать производительность

Каскадная репликация снижает нагрузку на мастер

Отложенная репликация позволяет вернуться назад на некоторый момент времени без использования архива

Поддерживаются самые разные конфигурации, из которых надо выбрать подходящую для задачи

1. Настройте каскадную репликацию между тремя серверами. Первый сервер — основной, второй (под пользователем `postgres2`) — промежуточная реплика, третий (под пользователем `postgres3`) — последняя реплика.
2. Настройте третий сервер на применение записей WAL с задержкой в десять секунд.
3. Проверьте работу репликации, убедитесь в том, что на третьем сервере данные появляются с установленной задержкой.
4. Остановите мастер и перейдите на второй сервер.
5. Проверьте, что третий сервер продолжает работать в режиме реплики.



Авторские права

Курс «Администрирование PostgreSQL 9.5. Расширенный курс»

© Postgres Professional, 2016 год.

Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь

Отзывы, замечания и предложения направляйте по адресу:

edu@postgrespro.ru

Отказ от ответственности

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Подключение нескольких реплик

Синхронная репликация

Каскадная репликация

Отложенная репликация

Задача

повысить доступность и распределить нагрузку

Механизм

несколько реплик подключаются к одному мастеру

Ограничения и особенности

на каждую реплику требуется отдельный процесс wal sender и отдельный слот репликации

Механизм репликации позволяет сконструировать систему так, чтобы она отвечала предъявляемым к ней требованиям. Рассмотрим несколько типичных задач и средства их решения.

Задача: обеспечить высокую доступность и распределение нагрузки.

Для решения необходимо иметь мастер-сервер и несколько реплик. Реплики можно использовать для выполнения запросов только на чтение; при сбое основного сервера можно перейти на реплику с минимальным временем простоя.

Несколько реплик можно подключить к мастер-серверу точно так же, как это рассматривалось в предыдущих темах на примере одной реплики. Надо только учитывать, что каждой реплике будет соответствовать отдельный процесс wal writer и отдельный слот репликации.

Несколько реплик

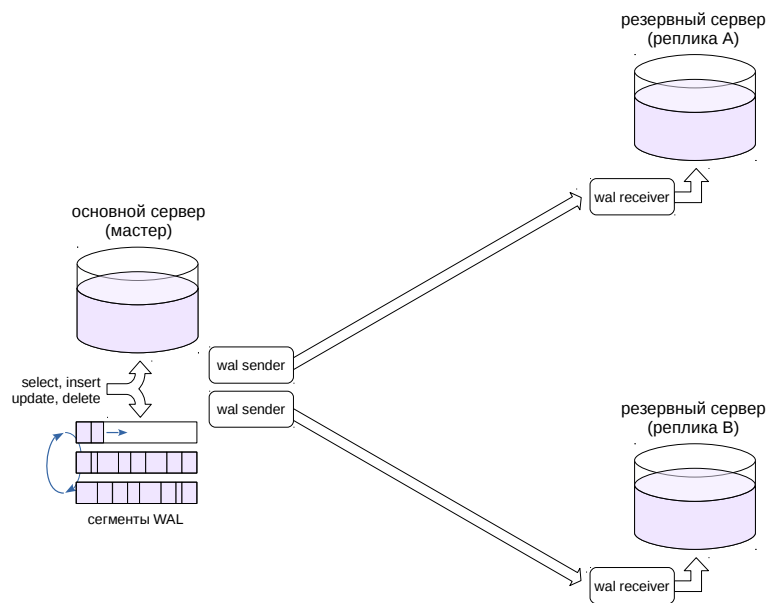


Иллюстрация подключения нескольких реплик к одному основному серверу.

Распределение нагрузки должно решаться внешними средствами, например, Pgpool-II.

Мастер

postgresql.conf
max_wal_senders
max_replication_slots

Реплика

дополнительные настройки не требуются

Для настройки требуется на мастере установить достаточные значения для параметров `max_wal_senders` и `max_replication_slots` по количеству реплик (плюс `pg_basebackup`, `pg_receivexlog` и т. п.).

На репликах никаких дополнительных настроек не требуется.

Задача

обеспечить надежность хранения данных

Механизм

при фиксации изменений мастер получает подтверждение от реплики, прежде чем вернуть управление

Ограничения

обеспечивается надежность, но не согласованность
синхронизация возможна только с одной репликой
при отключении одной синхронной реплики мастер начинает синхронизироваться со следующей из списка

Задача: в случае сбоя основного сервера, не потерять никакие данные при переходе на реплику.

Решение состоит в использовании синхронной репликации. В случае одного сервера есть понятие синхронной записи журнала упреждающей записи — это гарантирует, что данные не будут потеряны при сбое, если фиксация изменений успешно завершилась. Аналогичный механизм работает и для репликации: фиксация изменений на мастере не завершается до тех пор, пока не будет получено подтверждение от реплики.

Надо понимать, что синхронная репликация не обеспечивает идеальной согласованности данных между серверами: изменения могут становиться видимыми на мастере и на реплике в разные моменты времени.

Основной сервер хранит список реплик для синхронизации. Одновременно синхронизация происходит только с одной репликой. Если та становится недоступной, мастер пытается синхронизироваться со следующей из списка.

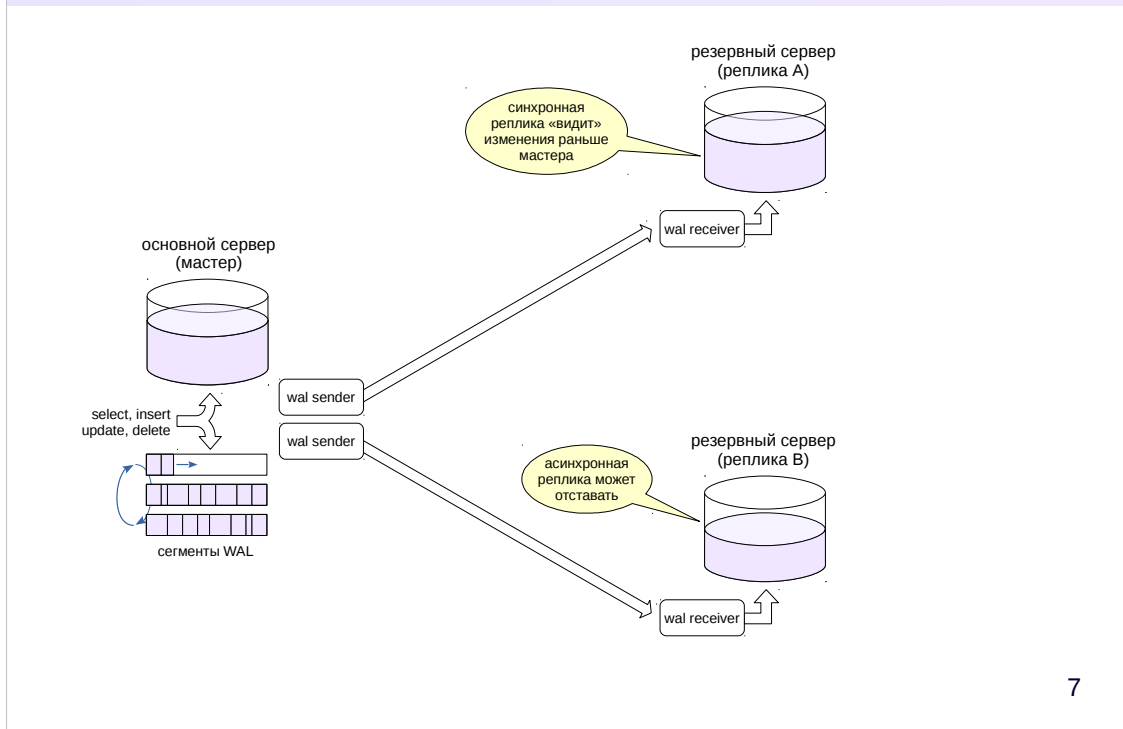


Иллюстрация синхронной репликации.

Реплика А синхронная. При фиксации изменений мастер выполняет следующее:

- делает запись в журнал упреждающей записи (таким образом, изменение не потеряется при сбое);
- дожидается подтверждения от реплики о получении записи WAL;
- изменяет состояние транзакции в буфере CLOG.

Таким образом, запросы на синхронной реплике могут увидеть изменения даже раньше, чем запросы на мастере.

Реплика Б асинхронная, она может отставать.

Мастер

postgresql.conf

`synchronous_standby_names`

список синхронных реплик (`application_name`) через запятую
звездочка (*) обозначает всех, но осторожнее с `pg_receivexlog`

`synchronous_commit`

`off, local` выключено

`remote_write` подтверждение записи в файл (в буфер ОС)

`on` подтверждение попадания файла на диск

Реплика

recovery.conf

`primary_conninfo = '... application_name=имя ...'`

Для настройки синхронной репликации на репликах, которые потенциально могут быть синхронными, надо в параметр `primary_conninfo` вписать имя приложения (`application_name`). По умолчанию оно будет равно «walreceiver» по имени процесса.

На основном сервере настраивается параметр `synchronous_standby_names`, содержащий список имен приложений. Вместо списка можно указать звездочку, которая говорит о том, что любая реплика может быть синхронной. Однако для PostgreSQL не отличимы реплики и другие программы, работающие по протоколу репликации (например, `pg_receivexlog`), поэтому при их использовании звездочка не годится.

Параметр `synchronous_commit` должен иметь значение `remote_write` (дождаться подтверждения записи в файл) или `on` (дождаться подтверждения физического попадания записи на диск). Второе дольше, но надежнее.

Особенности

- фиксация занимает время, производительность мастера падает
 - можно включать синхронизацию только для критичных транзакций
`set local synchronous_commit = on;`
 - синхронную реплику лучше не нагружать
- если ни одна реплика из списка синхронизации не подключена, фиксация изменений никогда не завершится
 - стоит иметь несколько реплик
- синхронная реплика содержит наиболее полный набор данных
 - именно на нее надо переключаться с мастера

Очевидно, что подтверждение от реплики существенно замедлит работу основного сервера. Поэтому как минимум следует не нагружать синхронную реплику запросами. Иногда транзакции можно разделить на «очень важные» (например, критичные для бизнеса) и «не очень важные» (вспомогательные): очень важные можно фиксировать, включая параметр синхронизации на уровне транзакции, а остальные могут работать в обычном режиме.

Одну — синхронную — реплику иметь небезопасно, так как при выходе ее из строя фиксация изменений никогда не завершится: основной сервер будет ждать подтверждения. Поэтому стоит иметь несколько реплик.

В случае перехода с основного сервера на реплику надо выбирать синхронную, так как именно она содержит все данные.

Задача

иметь несколько реплик, не нагружая мастер

Механизм

одна реплика передает полученные данные другой реплике

Ограничения и особенности

не поддерживается синхронная репликация

мастеру поступает обратная связь от всех реплик по цепочке

переключаться с мастера имеет смысл на ближайшую реплику

Задача: иметь несколько реплик, не создавая дополнительной нагрузки на основной сервер.

Задача решается с помощью каскадной репликации, при которой одна реплика передает записи WAL другой реплике и так далее.

При каскадной репликации не поддерживается синхронизация. Однако обратная связь поступает основному серверу от всех реплик, так что этот функционал работает в полном объеме.

При необходимости переключения следует выбирать ближайшую к мастеру реплику, как наименее запаздывающую.

Каскадная репликация

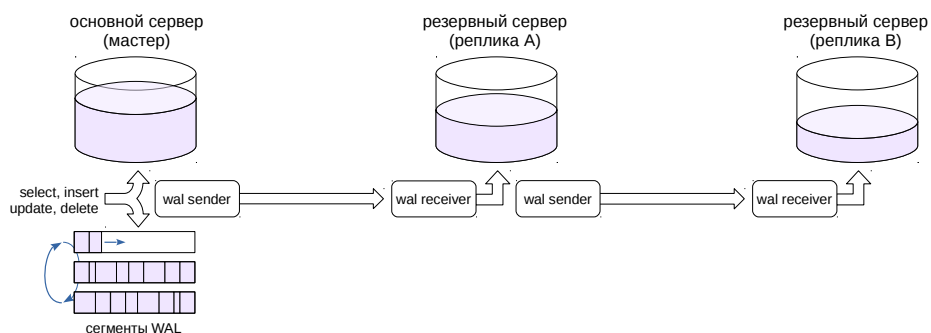


Иллюстрация каскадной репликации. На основном сервере только один процесс wal sender; реплики передают записи WAL друг другу по цепочке. Чем дальше от мастера, тем большее может накопиться запаздывание. Схема мониторинга усложняется: процесс надо контролировать на нескольких серверах.

Мастер

дополнительные настройки не требуются

Промежуточная реплика

postgresql.conf

max_wal_senders

max_replication_slots

pg_hba.conf

разрешение для базы replication

recovery.conf

recovery_target_timeline = 'latest'

Каскадная репликация не требует специфических настроек. На промежуточных репликах требуется установить достаточное значение параметров `max_wal_senders` и `max_replication_slots` (в любом случае лучше, чтобы эти значения совпадали со значениями на мастере) и убедиться, что следующая по цепочке реплика сможет подключиться к данной по протоколу репликации (настройки в `pg_hba.conf`).

Тонкий момент: если совершится переход на реплику, произойдет увеличение номера ветви времени. Чтобы остальные реплики продолжали получать изменения, для них параметр `recovery_target_timeline` должен быть явно указан и равен 'latest'.

Задача

«машина времени» и возможность восстановиться на определенный момент в прошлом без архива

Механизм

искусственная задержка применения записей WAL

Ограничения и особенности

нужна синхронизация часов между репликами
откладывается применение записей о фиксации (многоверсионность)
при окончании восстановления остаток применяется без задержки
включенная обратная связь при большом интервале приведет к увеличению размера таблиц на мастере из-за старых версий строк

Задача: иметь возможность просмотреть данные на некоторый момент в прошлом и, при необходимости, восстановить сервер на этот момент.

Проблема в том, что обычный механизм восстановления из архива на момент времени (point-in-time recovery) в принципе позволяет решить задачу, но требует большой подготовительной работы и занимает много времени. А механизм «отката» изменений (аналога undo в некоторых СУБД) в PostgreSQL не предусмотрен.

Задача решается созданием реплики, которая применяет записи WAL не сразу, а через установленный интервал времени.

Чтобы задержка работала правильно, необходима синхронизация часов между серверами.

Откладывается применение не всех записей, а только записей о фиксации изменений. Это означает, что часть записей (до записи о фиксации) может быть применена «раньше времени», но это не представляет проблемы благодаря использованию многоверсионности.

Если вывести реплику из режима непрерывного восстановления (с помощью `pg_ctl promote` или триггерного файла), остаток записей будет применен без каких-либо задержек.

При использовании обратной связи надо проявлять осторожность, поскольку большая задержка вызовет разрастание таблиц на мастере из-за того, что очистка не будет удалять старые версии строк, которые могут быть нужны реплике.

Отложенная репликация

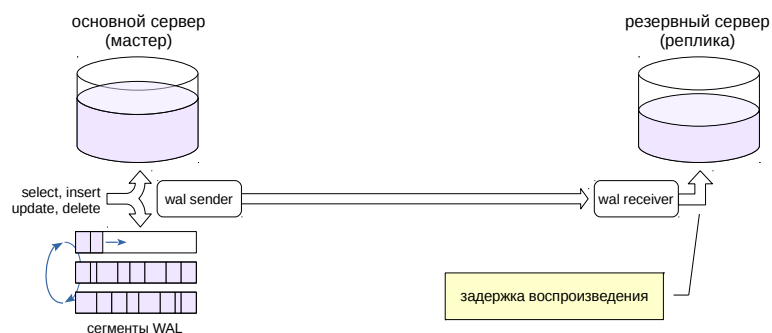


Иллюстрация отложенной репликации. Задержка происходит при применении записей WAL, таким образом на реплике будут скапливаться полученные, но еще не примененные записи.

Мастер

дополнительные настройки не требуются

Реплика

recovery.conf

`recovery_min_apply_delay`

а также функции:

`pg_xlog_replay_pause()`

`pg_xlog_replay_resume()`

Для настройки отложенной репликации на реплике требуется установить параметр `recovery_min_apply_delay`.

Типичный сценарий применения отложенной репликации: на основном сервере происходит какая-либо проблема (допустим, удалены критичные данные). На реплике данные еще есть, поскольку соответствующие записи WAL еще не применены. В этом случае надо остановить дальнейшее проигрывание записей с помощью функции `pg_xlog_replay_pause()`, пока идет исследование проблемы.

Допустим, принято решение вернуться на момент времени, предшествующий удалению. В таком случае надо отредактировать файл `recovery.conf` реплики (чтобы указать момент, на котором надо прервать восстановление и перейти в обычный режим) и перезапустить ее, чтобы изменение вступило в силу.

Несколько реплик увеличивают отказоустойчивость и позволяют распределять нагрузку

Синхронная репликация увеличивает надежность, но нужно учитывать производительность

Каскадная репликация снижает нагрузку на мастер

Отложенная репликация позволяет вернуться назад на некоторый момент времени без использования архива

Поддерживаются самые разные конфигурации, из которых надо выбрать подходящую для задачи

1. Настройте каскадную репликацию между тремя серверами. Первый сервер — основной, второй (под пользователем postgres2) — промежуточная реплика, третий (под пользователем postgres3) — последняя реплика.
2. Настройте третий сервер на применение записей WAL с задержкой в десять секунд.
3. Проверьте работу репликации, убедитесь в том, что на третьем сервере данные появляются с установленной задержкой.
4. Остановите мастер и перейдите на второй сервер.
5. Проверьте, что третий сервер продолжает работать в режиме реплики.