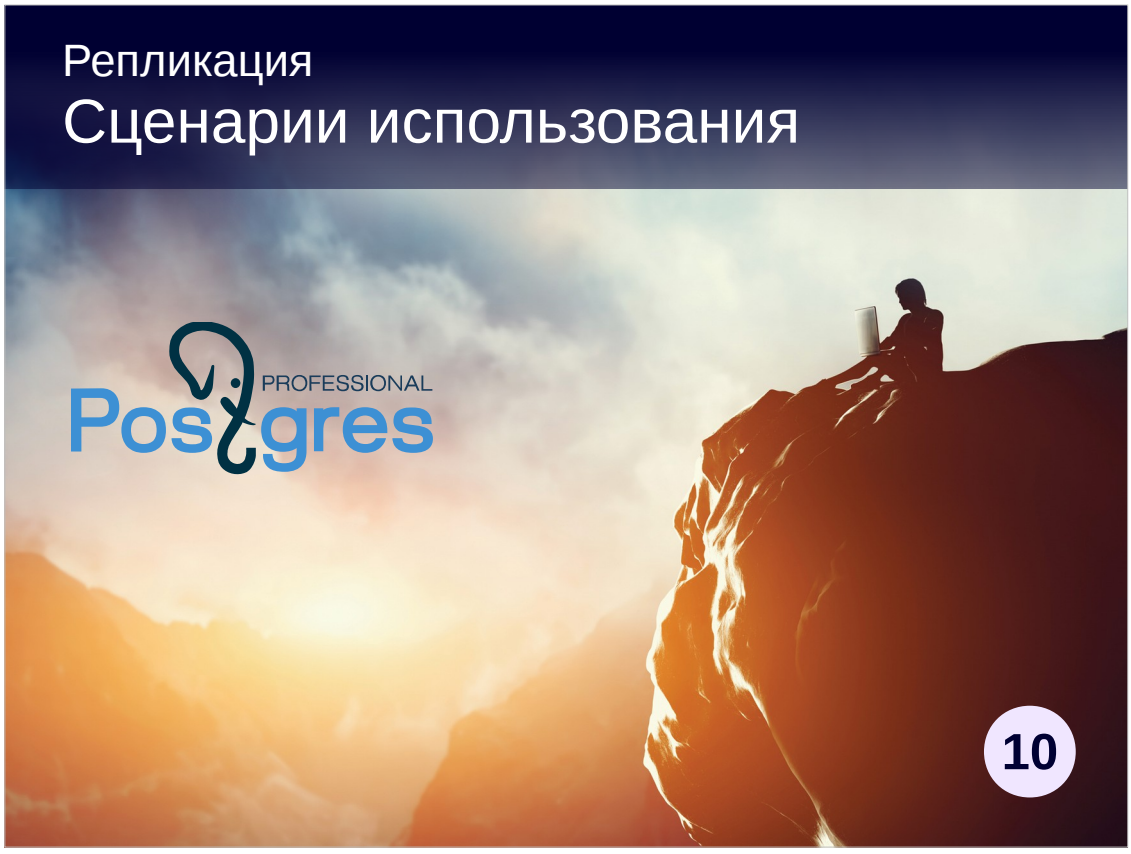


Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Использование физической репликации

Использование логической репликации

Физическая репликация

Горячий резерв для высокой доступности

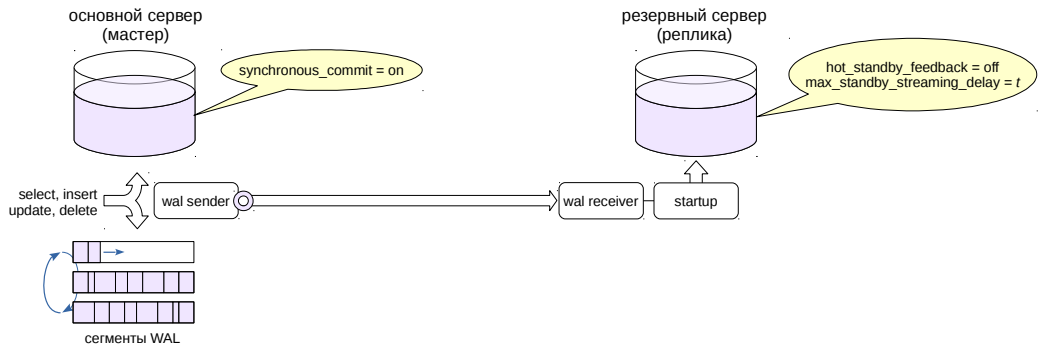
Балансировка OLTP-нагрузки

Реплика для отчетов

Несколько реплик и каскадная репликация

Отложенная репликация

Горячий резерв для HA



синхронная репликация: реплика максимально соответствует мастеру

запросы к реплике возможны, но не приоритетны:
они будут прерваны при любом конфликте

Реплика может создаваться для решения разных задач, и настройка зависит от ее предназначения.

Один из возможных вариантов — обеспечение горячего резерва для целей высокой доступности. Это означает, что при сбое основного сервера требуется как можно быстрее перейти на реплику, не потеряв при этом данные.

Надежность обеспечивает синхронная репликация в режиме *synchronous_commit* = on (значение write не гарантирует надежность; значение apply будет перебором).

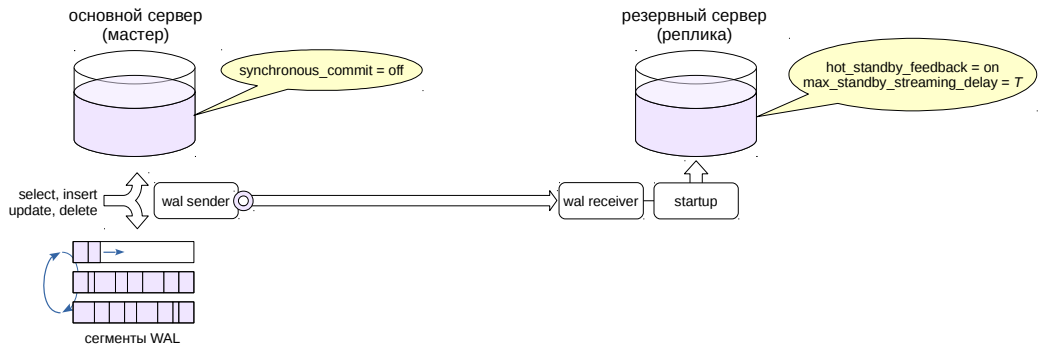
Чтобы реплика как можно меньше отставала от основного сервера, нужно применять журнальные записи сразу же. Для этого выставляем задержку *max_standby_streaming_delay* в небольшое значение.

Чтобы запросы к реплике не могли негативно повлиять на основной сервер, выключаем обратную связь (*hot_standby_feedback* = off).

С такими настройками запросы к реплике возможны, но в случае конфликтов они будут прерваны. Если запросы к реплике не предполагаются, можно создать ее в режиме «теплого резерва» (*hot_standby* = off).

Заметим, что репликация — базовый механизм для обеспечения высокой доступности, но далеко не все, что нужно. Более подробнее это обсуждается в модуле «Кластерные технологии».

Балансировка OLTP



запросы на реплике должны отрабатывать

долгие запросы повлияют на мастер, но в OLTP-нагрузке их не должно быть

согласованность данных должна обеспечиваться отдельно

Другой вариант — реплика используется для балансировки OLTP-нагрузки на чтение.

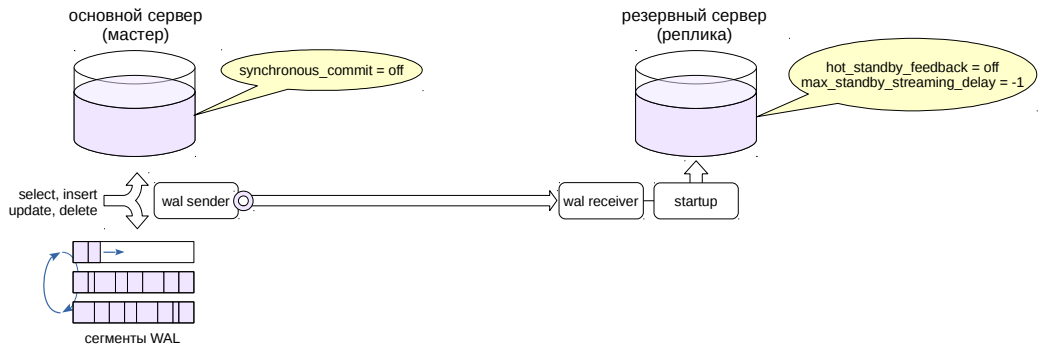
OLTP-нагрузка характеризуется небольшими по длительности запросами. Это позволяет достаточно тесно связать реплику с основным сервером, чтобы запросы гарантированно не прерывались из-за конфликтов, и в то же время не беспокоиться о негативном влиянии на основной сервер.

Включаем обратную связь (*hot_standby_feedback* = on) и выставляем достаточно большую задержку применение конфликтных записей (*max_standby_streaming_delay*).

Синхронная репликация в такой конфигурации, скорее всего, не требуется. Она все равно не обеспечивает согласованность данных: зафиксированное изменение на основном сервере не гарантирует, что запрос к реплике увидит эти изменения.

Для обеспечения согласованности нужно, чтобы изменения появлялись на обоих серверах *одновременно*. Режим *synchronous_commit* = apply позволяет дождаться применения журнальной записи на реплике (ценой сильного падения производительности основного сервера), но остается возможность увидеть данные на реплике раньше, чем на мастере. В версии 12 возможно появление специальной команды WAIT FOR LSN, которую можно будет выполнять на реплике для ожидания применения нужной журнальной записи (дискуссия). Надежное и прозрачное для приложений решение можно получить, используя специальные протоколы распределенных транзакций (см. модуль «Кластерные технологии»).

Реплика для отчетов



запросы на реплике должны отрабатывать, в том числе и долгие (но не требуются самые актуальные данные)

мастер не должен испытывать негативного влияния обратной связи

чтобы совсем развязать серверы — убрать слот и использовать архив WAL

Еще один возможный вариант — разделение по серверам нагрузки разного типа. Основной сервер может брать на себя OLTP-нагрузку, а длительные читающие запросы (отчеты) можно вынести на реплику.

Плюс такого решения состоит в том, что длительные запросы на основном сервере могут задерживать выполнение очистки и приводить к разрастанию таблиц.

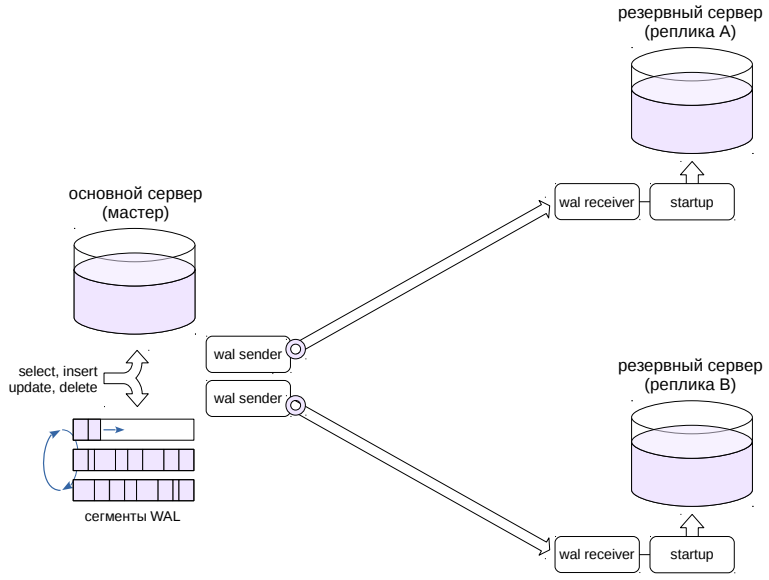
Из этих соображений «отчетная» реплика должна быть максимально отделена от основного серевра. Должна быть отключена обратная связь (*hot_standby_feedback* = off), но время откладывания конфликтующих журнальных записей нужно сильно увеличить, вплоть до бесконечности (*max_standby_streaming_delay* = -1).

Поскольку для длительных отчетов не требуются сиюминутные данные, можно, при наличии архива, отказаться от слота репликации (не забыв выставить параметр *max_standby_archive_delay* аналогично *max_standby_streaming_delay*). Тогда, если мастер успеет удалить необходимый реплике файл, реплика возьмет его из архива.

Такую реплику можно также использовать для выполнения резервного копирования.

Разумеется, на практике возможны и другие сочетания параметров. Главное — четко понимать, какую задачу требуется решить, и какое влияние оказывают те или иные параметры.

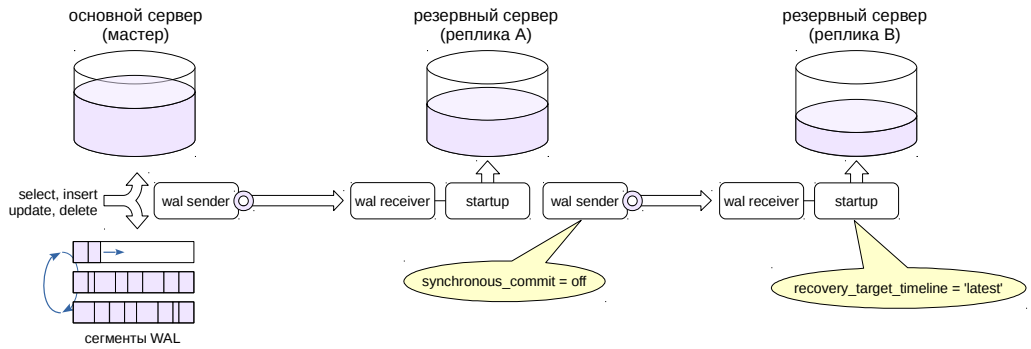
Несколько реплик



К основному серверу можно подключить несколько реплик. Никаких специальных настроек для этого не требуется, но надо учитывать, что каждой реплике будет соответствовать отдельный процесс wal sender и отдельный слот репликации.

Обычно такая схема требуется для распределения нагрузки, но собственно распределение должно решаться внешними средствами (см. модуль «Кластерные технологии»).

Каскадная репликация



позволяет снизить нагрузку на мастер и перераспределить сетевой трафик

не поддерживается каскадная синхронная репликация

обратная связь поступает от всех реплик

Несколько реплик, подключенных к одному основному серверу, будут создавать на него определенную нагрузку. Кроме того, надо учитывть нагрузку на сеть для пересылки нескольких копий потока журнальных записей.

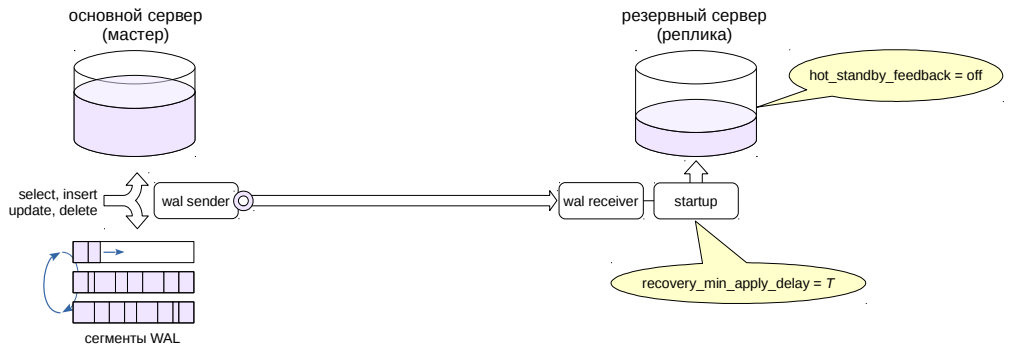
Для снижения нагрузки реплики можно соединять каскадом; при этом серверы передают журнальные записи друг другу по цепочке. Чем дальше от мастера, тем большее может накопиться запаздывание. Схема мониторинга усложняется: процесс надо контролировать на нескольких серверах.

Для настройки на промежуточных репликах требуется обеспечить достаточное значение параметров *max_wal_senders* и *max_replication_slots* и проверить настройки подключения по протоколу репликации в pg_hba.conf.

Тонкий момент: если совершится переход на реплику (ближайшую к основному серверу), то на ней произойдет увеличение номера ветви времени. Чтобы остальные реплики продолжали получать изменения, для них в recovery.conf нужно устанавливать параметр *recovery_target_timeline* = 'latest'.

Заметим, что каскадная синхронная репликация не поддерживается: основной сервер может быть синхронизирован только с непосредственно подключенной к нему репликой. А вот обратная связь поступает основному серверу от всех реплик.

Отложенная репликация



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

pg_wal_replay_pause()
pg_is_wal_replay_paused()
pg_wal_replay_resume()

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

Обычный механизм восстановления из архива на момент времени (point-in-time recovery) позволяет решить задачу, но требует большой подготовительной работы и занимает много времени.

Другое решение — создать реплику, которая применяет журнальные записи не сразу, а через установленный интервал времени (*recovery_min_apply_delay* в recovery.conf). Чтобы задержка работала правильно, необходима синхронизация часов между серверами.

Откладывается применение не всех записей, а только записей о фиксации изменений. Записи до фиксации могуть быть применены «раньше времени», но это не представляет проблемы благодаря многоверсионности.

Обратную связь следует отключать, чтобы не вызвать разрастание таблиц на мастере.

Типичный сценарий: на основном сервере происходит какая-либо проблема (допустим, удалены критичные данные). На реплике данные еще есть, поскольку соответствующие записи еще не применены. Дальнейшее проигрывание записей приостанавливается с помощью функции pg_wal_replay_pause(), пока идет исследование проблемы.

Если принято решение вернуться на момент времени до удаления, надо отредактировать файл recovery.conf (чтобы указать целевую точку восстановления) и перезапустить реплику.

Логическая репликация

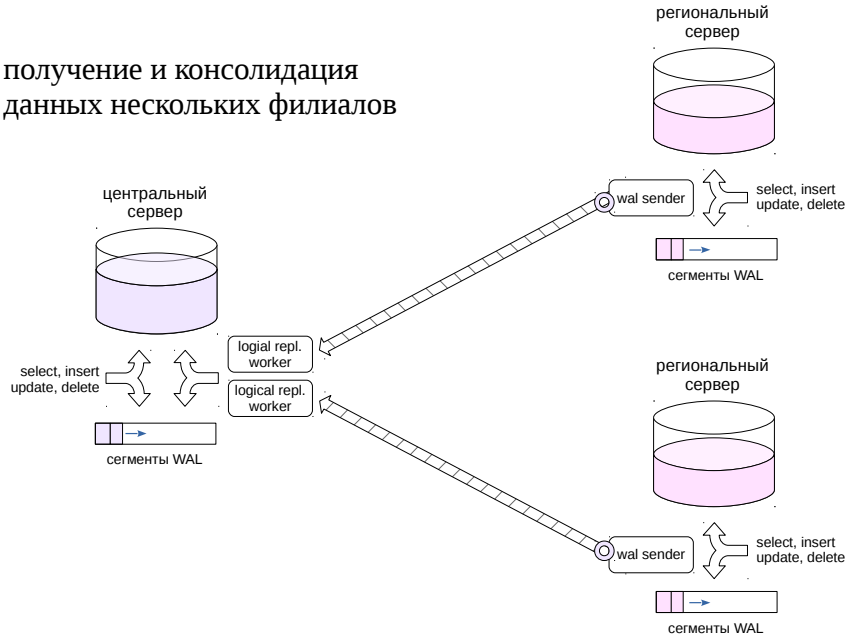
Консолидация и общие справочники

Обновление основной версии сервера

Мастер-мастер (в будущем)

Консолидация

получение и консолидация данных нескольких филиалов



Пусть имеются несколько региональных филиалов, каждый из которых работает на собственном сервере PostgreSQL. Задача состоит в консолидации части данных на центральном сервере.

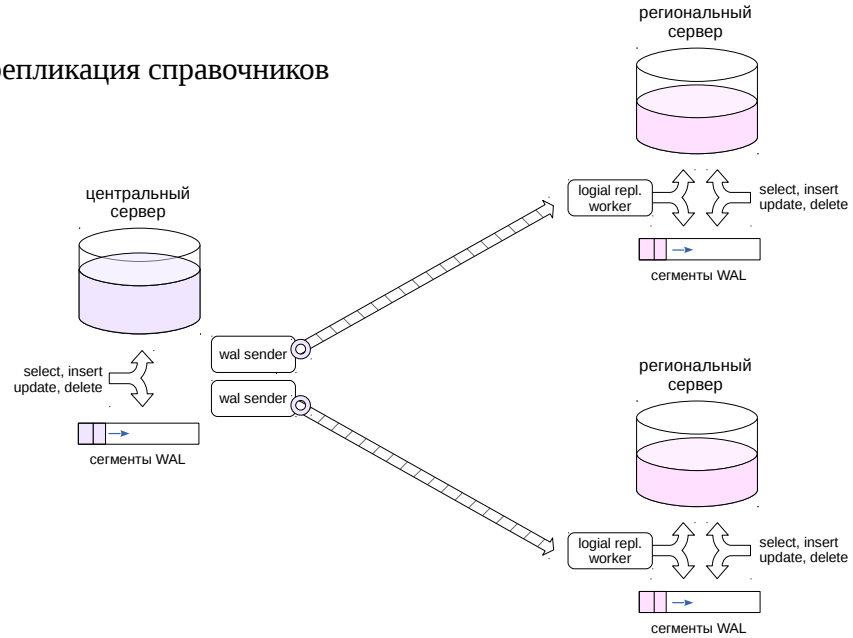
Для решения на региональных серверах создаются публикации необходимых данных. Центральный сервер подписывается на эти публикации. Полученные данные можно обрабатывать (например, приводить к единому виду) с помощью триггеров на стороне центрального сервера.

Поскольку репликация основана на передаче данных через слот, между серверами необходимо более или менее постоянное соединение, так как во время разрыва соединения региональные серверы будут вынуждены сохранять файлы журнала.

Есть множество особенностей такого процесса и с точки зрения бизнес-логики, требующих всестороннего изучения. В ряде случаев может оказаться проще передавать данные пакетно раз в определенный интервал времени.

Общие справочники

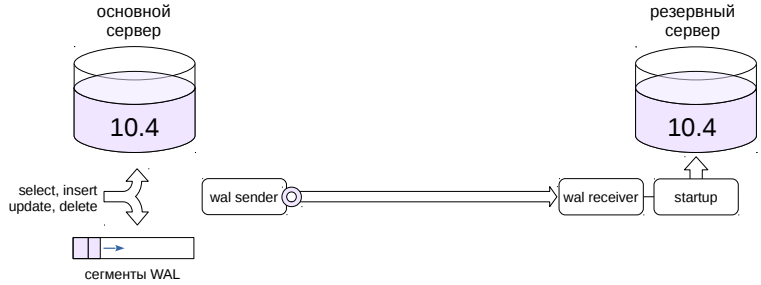
репликация справочников



Другая задача: на центральном сервере поддерживаются справочники, актуальные версии которых должны быть доступны на региональных серверах.

В этом случае схему надо развернуть наоборот: центральный сервер публикует изменения, а региональные серверы подписываются на эти обновления.

Обновление серверов

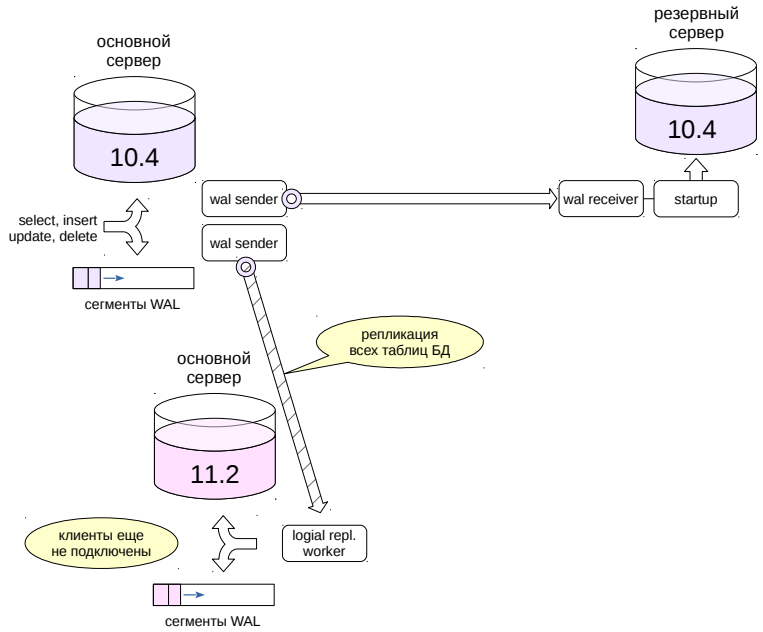


мастер и физическая реплика: требуется обновить основную версию

Логическая репликация может использоваться для обновления основной версии сервера без прерывания обслуживания (или с минимальным прерыванием).

Допустим, имеется основной сервер и физическая потоковая реплика.

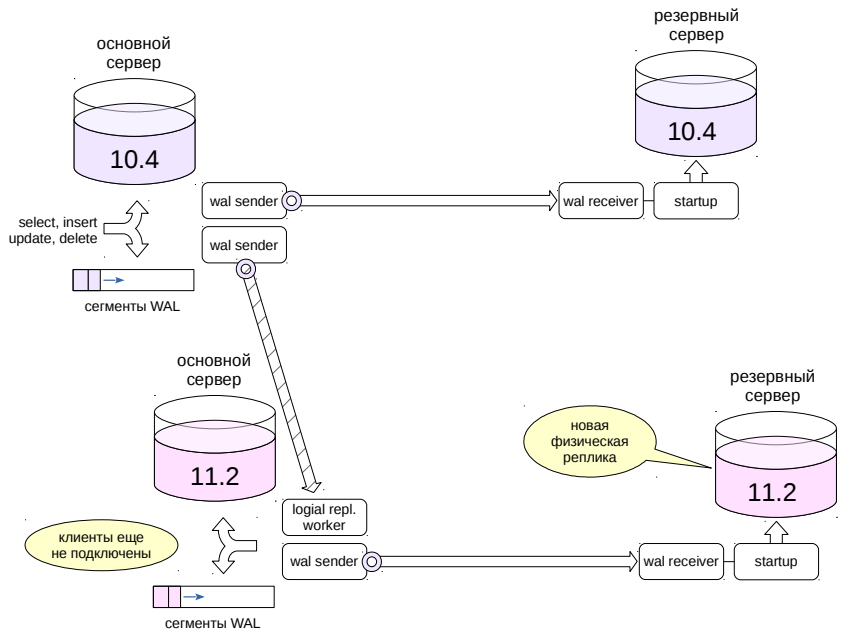
Обновление серверов



Создаем новый сервер с требуемой версией PostgreSQL и переносим на него структуру всех таблиц выбранной базы данных.

На основном сервере предыдущей версии публикуем изменения всех таблиц базы данных. Поскольку изменения схемы данных не реплицируются, на время обновления такие изменения должны быть запрещены.

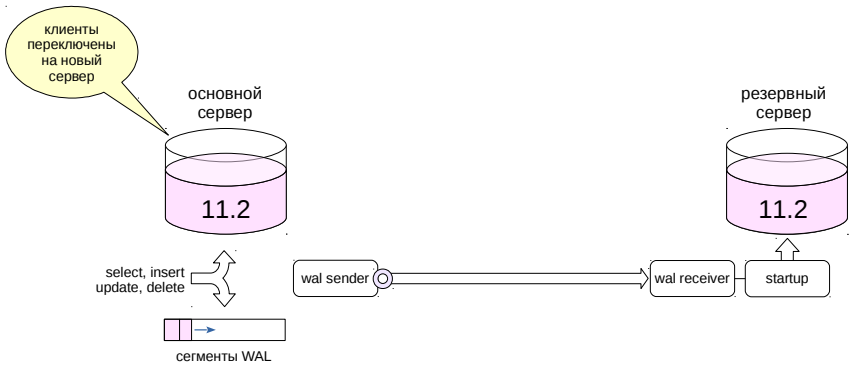
Обновление серверов



Создаем физическую реплику, аналогичную реплике сервера предыдущей версии.

В итоге получаем параллельную структуру серверов с новой версией PostgreSQL. Эта структура аналогочна имеющейся для старой версии.

Обновление серверов

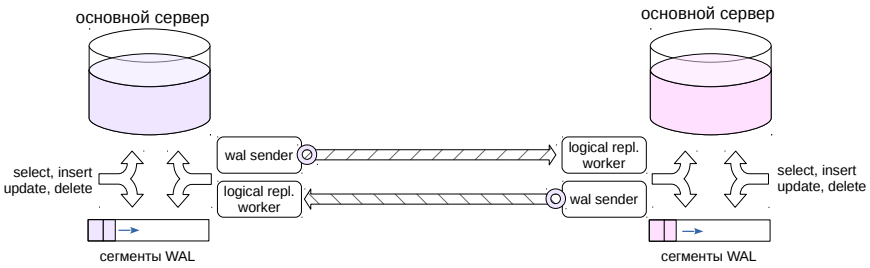


После этого переключаем клиентов на новые серверы, а старые останавливаем. Прерывание обслуживания будет определяться тем, насколько плавно можно выполнить это переключение.

Надо иметь в виду, что такой процесс нельзя считать универсальным: он накладывает достаточно много ограничений и существенно более сложен, чем использование утилиты pg_upgrade, которая позволяет выполнить обновление за небольшое время.

Мастер-мастер

кластер, в котором данные могут изменять несколько серверов
(дело будущего)



Задача: обеспечить надежное хранение данных на нескольких серверах с возможностью записи на любом сервере (что полезно, например, для георасределенной системы).

Для решения можно использовать двунаправленную репликацию, передавая изменения в одних и тех же таблицах от одного сервера к другому и обратно.

Сразу заметим, что в настоящее время средства, доступные в PostgreSQL 10, не позволяют этого реализовать, но со временем (не раньше 12-й версии, а скорее всего позже) эта возможность должна будет появиться в ядре (см. расширения pg_logical и BDR).

Конечно, прикладная система должна быть построена таким образом, чтобы избегать конфликтов при изменении данных в одних и тех же таблицах. Например, использовать глобальные уникальные идентификаторы или гарантировать, что разные серверы работают с разными диапазонами ключей.

Надо учитывать, что система мастер-мастер, построенная на логической репликации, не обеспечивает сама по себе выполнение глобальных распределенных транзакций. При использовании синхронной репликации можно гарантировать надежность, но не согласованность данных между серверами. Кроме того, никаких средств для автоматизации обработки сбоев, подключения или удаления узлов из кластера и т. п. в PostgreSQL не предусмотрено — эти задачи должны решаться внешним средствами.

Итоги

Физическая репликация — базовый механизм для решения целого ряда задач

- обеспечение высокой доступности
- балансировка однотипной нагрузки
- разделение разных видов нагрузки
- и др.

Логическая репликация расширяет возможности

Практика

1. Настройте каскадную репликацию между тремя серверами.
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.

1. Используйте серверы alpha, beta и gamma.