Авторы: Егор Рогов, Павел Лузанов

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

Темы

Понятие физической резервной копии

Холодное резервирование

Горячее резервирование

Физическая копия

Базовая физическая копия —
копия файловой системы кластера

+ быстрее, чем логическое резервирование
+ восстанавливается статистика
− можно восстановиться только на совместимой системе и на той же самой основной версии PostgreSQL
− выборочная копия невозможна, копируется весь кластер

Физическое резервирование подразумевает копирование всех файлов, относящихся к кластеру БД, то есть создание полной двоичной копии.

Копирование файлов работает быстрее, чем выгрузка SQL-команд при логическом резервировании; запустить сервер из созданной физической копии — дело нескольких минут, в отличие от восстановления из логической копии. Кроме того, нет необходимости заново собирать статистику — она также восстанавливается из копии.

Но есть и минусы. Из физической резервной копии можно восстановить систему только на совместимой платформе (та же ОС, та же разрядность, тот же порядок байтов в представлении чисел и т. п.) и только на той же основной версии PostgreSQL. Кроме того, невозможно сделать физическую копию отдельных баз данных кластера, возможно копирование только всего кластера целиком.

Холодное резервирование

Резервная копия остановленного сервера

Холодное резервирование

Выключенный сервер

если выполнена контрольная точка, то нужна только копия ФС

копию можно развернуть на другом сервере, независимо от PGDATA

\+ простота

− требуется прерывание обслуживания

Снимок файловой системы

как при неаккуратном выключении: при старте потребуется восстановление, но в копию войдут все нужные файлы журнала

данные, в т. ч. табличные пространства, должны войти в один снимок

\+ не надо останавливать сервер

− файловые системы, поддерживающие снимки, работают медленней

Смысл холодного резервирования состоит в том, чтобы сделать копию файловой системы в тот момент, когда она содержит согласованные данные. Восстановление из такой копии происходит просто: файлы разворачиваются, запускается сервер — и он сразу же готов к работе.

К сожалению, единственный вариант сделать такую копию — аккуратно (с выполнением контрольной точки) остановить сервер. Минус понятен: необходима остановка сервера (которая к тому же может оказаться длительной при большом объеме данных).

Время простоя можно сократить за счет предварительного выполнения rsync (или аналогичного инструмента) при работающем сервере. Тогда после останова сервера rsync докопирует только изменения (которых, предположительно, будет не много). Но простоя все равно не избежать.

Другой вариант — сделать копию несогласованных данных. Такая ситуация может возникнуть при неаккуратном отключении сервера или при создании снимка файловой системы (если ФС имеет такую возможность, и если все необходимые файлы попадают в один снимок).

В этом случае восстановление происходит аналогично, но при старте серверу потребуется выполнить восстановление согласованности. Это обычная автоматическая процедура восстановления после сбоя. Она не представляет проблемы, так как необходимые файлы журнала гарантированно попадут в копию, но она потребует некоторого времени.

https://postgrespro.ru/docs/postgresql/10/backup-file.html

Горячее резервирование

Резервная копия работающего сервера

Протокол репликации и слоты

Штатный инструмент — утилита pg_basebackup

Низкоуровневый API

Горячее резервирование

Работающий сервер

не просто несогласованные, но и динамически изменяющиеся данные
(проблема неатомарного чтения и записи)

для восстановления согласованности необходимы журнальные записи
от последней контрольной точки за все время копирования
(в процессе копирования сервер может удалить часть файлов)

\+ не требуется прерывание обслуживания

− нужны специальные инструменты

Горячее резервирование выполняется на работающем сервере, поэтому в копию совершенно точно попадут несогласованные данные.

Более того. При резервном копировании данные читаются не через буферный кэш, а напрямую из файлов. Содержимое файлов на диске, очевидно, изменяется во время копирования, а файловая система обычно не гарантирует атомарность чтения/записи 8-килобайтной страницы PostgreSQL. Поэтому в резервную копию будут попадать «безнадежные» страницы, к которым даже нельзя применить журнальные записи. Это первая сложность.

Вторая сложность состоит в том, что копирование файлов данных может занимать достаточно много времени. Но сервер, после выполнения очередной контрольной точки, может удалить часть файлов журнала, которые уже не нужны ему для восстановления после сбоя, но нужны для резервной копии.

Если не принять специальных мер, сделанная резервная копия будет непригодна для восстановления. Поэтому для горячего резервирования требуются специальные инструменты. PostgreSQL предоставляет низкоуровневый интерфейс, используя который можно реализовать надежное копирование. Этот интерфейс использует и штатная утилита pg_basebackup, и другие сторонние средства резервного копирования.

https://postgrespro.ru/docs/postgresql/10/continuous-archiving.html#BACKUP-BASE-BACKUP

Протокол репликации

Задачи

управление резервным копированием и репликацией
в частности, получение потока журнальных записей

Обслуживается процессом wal_sender

max_wal_senders
wal_level = replica или logical, но не minimal

Подключение

роль с атрибутом REPLICATION или SUPERUSER
разрешение на подключение в pg_hba.conf

Для упрощения задачи, сервер PostgreSQL предоставляет протокол репликации — специальный протокол для управления как собственно репликацией (рассматривается в одноименном модуле), так и резервным копированием. В частности, он позволяет получать поток журнальных записей, которые генерирует сервер.

На сервере подключение по протоколу репликации обслуживается процессом wal sender. Он похож на обычный обслуживающий процесс, который запускается при обычном подключении клиента, но понимает не SQL, а специальные команды. Число одновременно работающих процессов wal_sender ограничено значением параметра max_wal_senders.

Уровень журнала должен быть не ниже, чем replica (archive в версиях до 9.6). Дело в том, что на уровне minimal такие команды, как CREATE TABLE AS SELECT, CREATE INDEX, COPY FROM, не попадают в журнал: их долговечность обеспечивается тем, что данные не остаются в оперативной памяти, а сразу записываются на диск. Этого достаточно для восстановления после сбоя и из холодной копии, но недостаточно для восстановления из горячей копии.

Чтобы использовать протокол репликации, клиент должен подключаться к серверу под ролью, имеющей атрибут REPLICATION (либо под суперпользователем). В pg_hba.conf надо разрешить подключение этой роли к базе данных replication (это, конечно, не название БД, а ключевое слово). Причем разрешения для all недостаточно, replication должен быть разрешен отдельно.

https://postgrespro.ru/docs/postgresql/10/protocol-replication

Слот репликации

Серверный объект для получения журнальных записей

- помнит, какая запись была передана клиенту последней
- число слотов ограничено max_replication_slots

Если используется слот, то сегмент WAL не удаляется, пока все его записи не переданы клиенту

- позволяет клиенту получать данные в удобном режиме, в том числе отключаться на время
- при отключении журнальные файлы будут накапливаться на сервере
- мониторинг состояния: pg_replication_slots

Чтобы сервер не удалил необходимые файлы WAL преждевременно, можно применять слот репликации. Если поток журнальных записей идет через слот, то слот помнит, какие записи уже были переданы клиенту. Наличие слота не позволит серверу удалять файл WAL до тех пор, пока клиент не получит все записи из этого файла.

Использование слота позволяет клиенту не беспокоиться о том, что сервер сотрет файл журнала раньше времени. Клиент даже может отключиться и затем через какое-то время подключиться вновь и продолжить получать журнальные записи с того момента, на котором остановился.

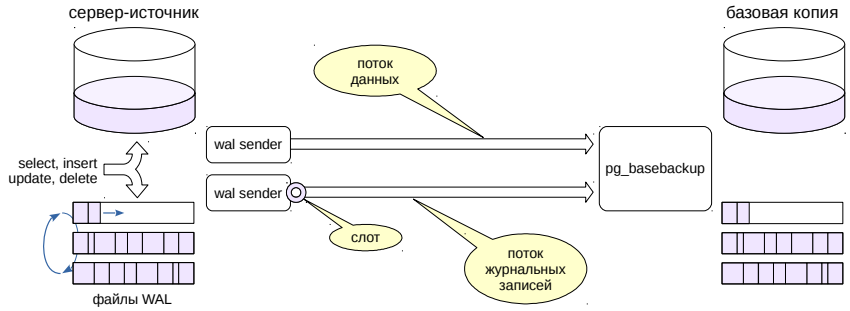
Но надо иметь в виду, что при отключении клиента файлы журнала будут накапливаться на сервере и могут занять все свободное место. Поэтому каждый созданный слот следует добавлять в мониторинг (представление pg_replication_slots) и своевременно удалять ненужные слоты.

Общее количество слотов, которые могут быть созданы, ограничено конфигурационным параметром wax_replication_slots.

Более подробно применение слотов рассматриваться в модуле «Репликация».

https://postgrespro.ru/docs/postgresql/10/warm-standby.html#STREAMING-REPLICATION-SLOTS

pg_basebackup



два потока, max_wal_senders ≥ 2

по умолчанию используется временный слот репликации

Для выполнения копирования утилита pg_basebackup использует два подключения по протоколу репликации: одно для передачи данных и одно — для передачи потока журнальных записей, которые генерирует работающий сервер во время копирования. Поэтому для pg_basebackup параметр max_wal_senders должен быть равен не менее 2.

Для передачи журнальных записей pg_basebackup, начиная с 10-й версии, PostgreSQL по умолчанию использует временный слот репликации, который существует только на время соединения и удаляется при завершении работы pg_basebackup.

Однако в параметрах утилиты можно указать и имя обычного слота.

https://postgrespro.ru/docs/postgresql/10/app-pgbasebackup

Цели и форматы

Немедленное развертывание нового экземпляра

`--format=plain`

удаленный запуск на сервере, где будет развернут экземпляр

копирует файлы и каталоги кластера (PGDATA) в указанный каталог

копирует табличные пространства по тем же абсолютным путям, но можно сопоставить и другие пути (`--tablespace-mapping`)

Резервная копия для последующего использования

`--format=tar`

`--gzip` или `--compress=0..9` для сжатия

удаленный или локальный запуск

помещает PGDATA в base.tar, файлы журнала в pg_wal.tar

помещает каждое табличное пространство в отдельный файл *OID*.tar, пути могут быть изменены в файле tablespace_map

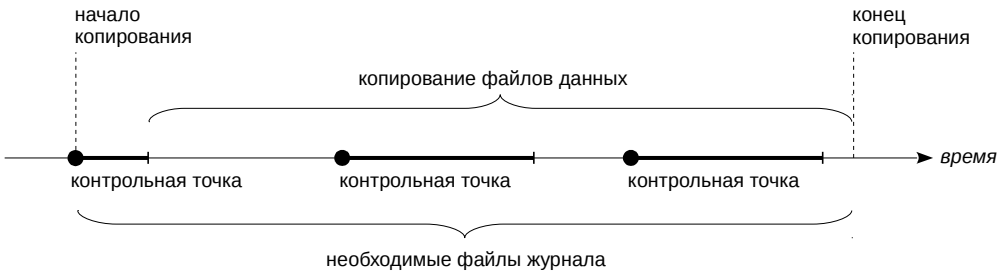
Если предполагается немедленно развернуть новый сервер из резервной копии, удобно вызывать pg_basebackup с форматом plain (используется по умолчанию), запуская его на целевом сервере. Утилита удаленно подключается к серверу-источнику и создает локальные каталоги и файлы, соответствующие каталогам и файлам основного сервера. Таким образом, новый сервер можно запускать, как только отработает pg_basebackup.

Табличные пространства будут скопированы по тем же абсолютным путям, что и на сервере-источнике (поэтому в таком режиме pg_basebackup нельзя запускать на сервере-источнике). Однако при необходимости можно переназначить пути для табличных пространств, указав соответствие в параметрах утилиты.

Если же копия выполняется в рамках обычной политики резервного копирования, удобно воспользоваться форматом tar. В этом случае pg_basebackup можно запускать как на сервере-источнике, так и удаленно. Основной каталог кластера PGDATA будет сохранен в файле base.tar, журналы — в файле pg_wal.tar, а табличные пространства — каждое в своем собственном tar-файле, имя которого будет совпадать с OID табличного пространства. Файлы могут быть сжаты, если указать соответствующие ключи утилиты.

Для восстановления из такой копии сначала потребуется развернуть tar-файлы по правильным путям. При этом табличные пространства можно разместить по новым путям, но потребуется отредактировать файл tablespace_map перед запуском сервера.

Низкоуровневый взгляд



для борьбы с неатомарностью чтения и записи

на время копирования устанавливается full_page_writes = on

в самом начале выполняется контрольная точка (быстро или обычно)

Общий алгоритм изготовления резервной копии на низком уровне одинаков как для pg_basebackup, так и для любых сторонних средств (которые могут потребоваться, поскольку pg_basebackup предоставляет только самую базовую функциональность).

1. Надо сообщить серверу о том, что начинается резервное копирование.

При этом, во-первых, на время копирования устанавливается параметр full_page_writes: при первом изменении страницы после контрольной точки полный образ этой страницы записывается в журнал. При восстановлении журнальные записи будут применяться не к страницам в файле (которые, как мы видели, могут быть прочитаны в рассогласованном состоянии), а к образу страницы из журнала.

Во-вторых, выполняется контрольная точка. Предусмотрено два режима: быстрое выполнение (что может привести к пиковой нагрузке на дисковую подсистему) и протяженное (которое определяется обычным параметром checkpoint_completion_target).

2. После прохождения контрольной точки можно копировать файлы данных любым удобным способом.

3. После того, как все скопировано, надо сообщить серверу, что резервное копирование завершено.

4. Кроме того, так или иначе надо обеспечить попадание в резервную копию всех журнальных записей, сгенерированных с начала копирования и до его окончания.

API резервирования

Начало копирования: pg_start_backup

- монопольный и немонопольный режимы
- выполняет контрольную точку, устанавливает full_page_writes
- возвращает начальную позицию в журнале

Конец копирования: pg_stop_backup

- возвращает конечную позицию в журнале

Копирование файлов

- либо репликационный протокол, либо любые средства ОС

Копирование журнала от начальной до конечной позиции

- либо репликационный протокол, либо средства ОС + wal_keep_segments

Описанный алгоритм резервного копирования можно использовать, чтобы реализовать свою собственную утилиту копирования.

Начало резервного копирования выполняется функцией pg_start_backup. Она устанавливает параметр full_page_writes и выполняет контрольную точку, как рассматривалось выше. Возвращаемое этой функцией значение — позиция в журнале, начиная с которой в копию должны попасть журнальные записи.

Начать резервное копирования можно в монопольном режиме (он чуть проще, но считается устаревшим) либо в немонопольном (позволяет делать несколько резервных копий одновременно).

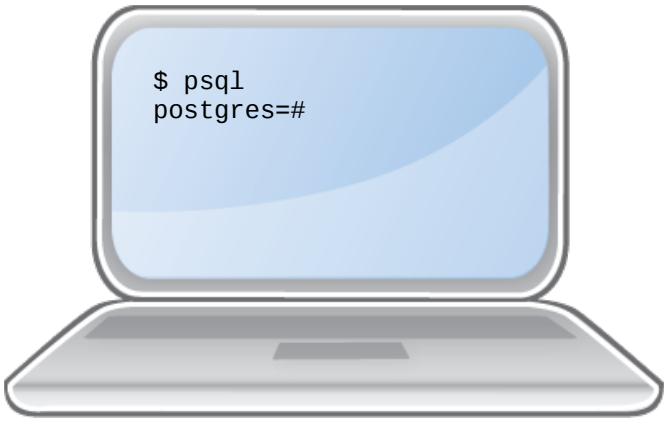
Конец копирования отмечается функцией pg_stop_backup. Она возвращает конечную позицию в журнале.

Все журнальные записи между начальной и конечной позициями надо поместить в резервную копию. Поскольку при своей работе сервер может удалять журнальные файлы, которые не требуются для восстановления, надо либо пользоваться протоколом репликации и слотом, получая записи во время копирования (как pg_basebackup), либо (что менее удобно) копировать файлы средствами ОС, установив параметр wal_keep_segments.

Копирование файлов данных также можно выполнить через протокол репликации (как pg_basebackup), так и средствами ОС. Можно копировать файлы в несколько потоков, использовать rsync и т. п.

https://postgrespro.ru/docs/postgresql/10/continuous-archiving.html#BACKUP-LOWLEVEL-BASE-BACKUP

Демонстрация



Итоги

Базовая резервная копия —
полная копия кластера, включая табличные пространства, плюс журнал для восстановления согласованности

Горячее резервирование позволяет не останавливать сервер

Инструменты

- протокол репликации
- слот репликации
- низкоуровневый API резервирования

Практика

1. В первом кластере создайте табличное пространство и базу данных с таблицей в этом пространстве.
2. Сделайте базовую резервную копию кластера с помощью pg_basebackup в формате tar со сжатием.
3. Разверните второй кластер из этой резервной копии. Табличное пространство разместите в другом каталоге, изменив файл tablespace_mapping.
4. Запустите второй сервер и проверьте его работоспособность.
5. Удалите базу данных и табличное пространство в обоих кластерах.