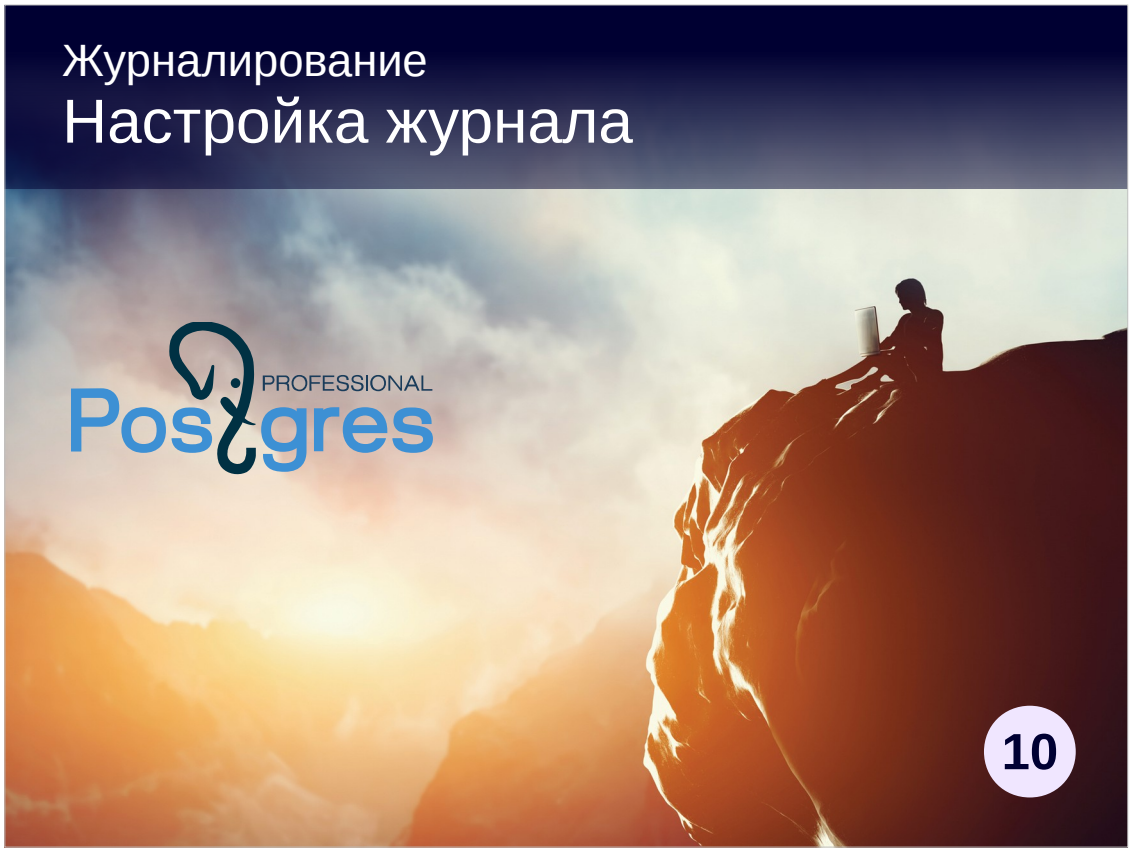


Авторы: Егор Рогов, Павел Лузанов

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

Темы

Уровни журнала и решаемые задачи

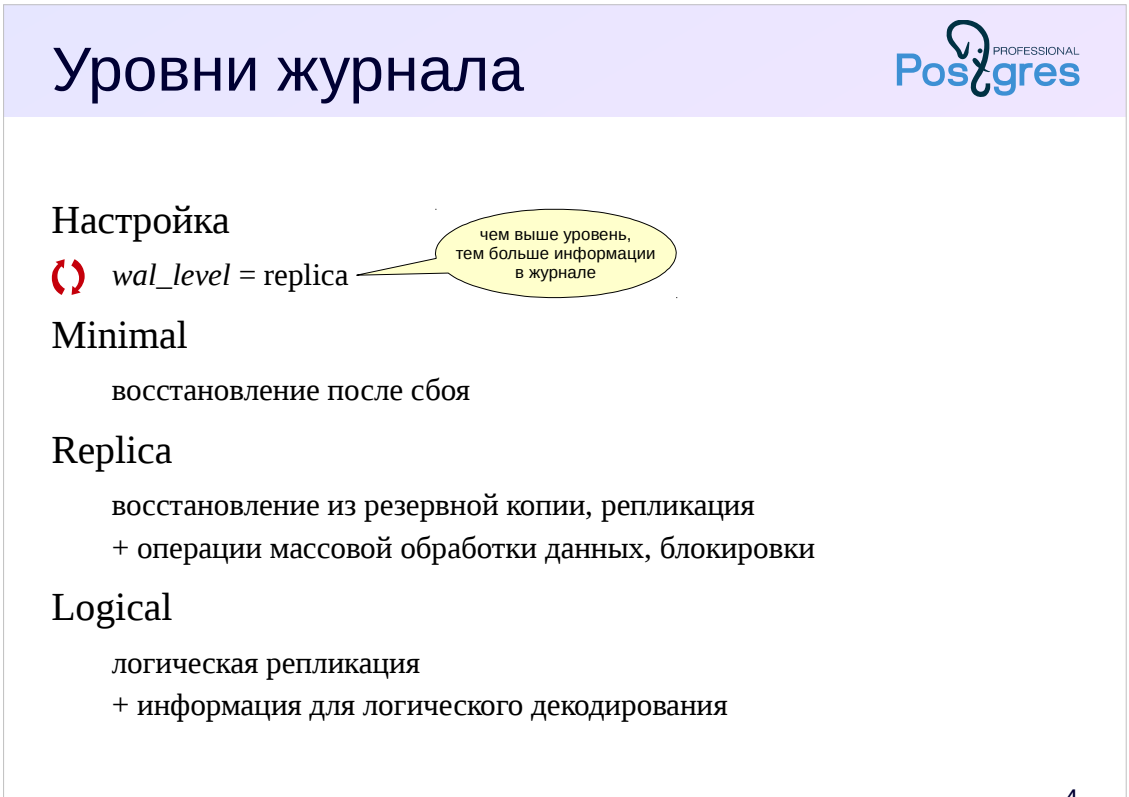
Надежность записи

Производительность

Уровни журнала

minimal, replica, logical

Уровни журнала



Кроме основной задачи — восстановления согласованности после сбоя — журнал можно и удобно использовать и для других целей, если добавить в него дополнительную информацию. Для этого существуют несколько уровней журнала, устанавливаемые параметром *wal_level*.

До версии PostgreSQL 10 уровнем по умолчанию был **minimal,** обеспечивающий только восстановление после сбоя.

Сейчас уровень по умолчанию — **replica** (до версии 9.6 было два отдельных уровня archive и hot_standby, которые объединили в один).

Этот уровень позволяет восстанавливать систему из горячих резервных копий, сделанных утилитой pg_basebackup, а также использовать репликацию — передавать поток журнальных записей на другой сервер (эти темы рассматриваются в курсе DBA3).

В журнал дополнительно записываются операции массовой обработки данных (такие, как CREATE TABLE AS SELECT, CREATE INDEX и т. п.). В минимальном режиме этого не происходит, а долговечность гарантируется немедленной записью данных на диск. Для репликации записывается информация о ряде блокировок.

Уровень **logical** используется для логической репликации — он должен быть включен на сервере, публикующем изменения.

Для правильной работы логического декодирования в журнал записывается дополнительная информация, позволяющая корректно расшифровать смысл операций по журнальным записям.

Чем выше уровень, тем больше информации попадает в журнал.

Надежность записи

Кэширование

Повреждение данных

Неатомарность записи страниц

Кэширование

Синхронизация с диском

данные должны дойти до энергонезависимого хранилища через многочисленные кэши

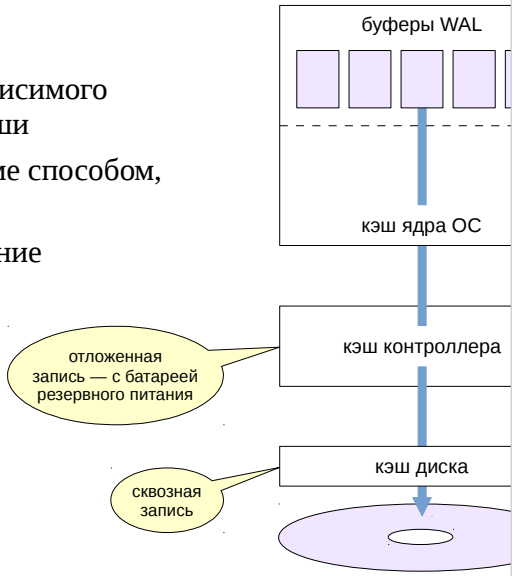
СУБД сообщает операционной системе способом, указанным в *wal_sync_method*

надо учитывать аппаратное кэширование

Настройки

fsync = on

wal_sync_method
(утилита pg_test_fsync)



Есть несколько моментов, которые влияют на надежность. Во-первых, всевозможные кэши, находящиеся на пути к энергонезависимому хранилищу (такому, как пластина жесткого диска).

Когда PostgreSQL требуется надежно записать данные, он пользуется способом, указанным в параметре *wal_sync_method*. Их несколько, но основных варианта два: либо после записи операционной системе дается команда синхронизации (fsync, fdatasync), либо файл открывается или записывается со специальным флагом, который говорит о необходимости синхронизации, а по возможности — прямой записи, минуя кэш ОС. Утилита pg_test_fsync позволяет выбрать способ, наиболее подходящий для конкретной ОС и конкретной файловой системы. Но в любом случае это дорогостоящая операция.

Кроме этого, администратор должен убедиться в том, что данные действительно доходят до диска, а не задерживаются в кэше контроллера или самого диска. Если кэш контроллера откладывает запись, то контроллер в обязательном порядке должен иметь батарею резервного питания. Для дисков лучше устанавливать сквозную запись.

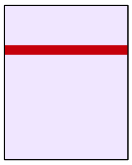
Вообще говоря, синхронизацию можно отменить (параметр *fsync*), но в этом случае про надежность хранения следует забыть. Единственный разумный вариант отключения этого параметра — временное увеличение производительности в случае, если данные можно легко восстановить (например, при начальной миграции).

https://postgrespro.ru/docs/postgresql/10/wal-reliability

Повреждение данных

Контрольные суммы журнальных записей

включены всегда, CRC-32



Контрольные суммы страниц

нужно включить при инициализации кластера

```
initdb -k
```

накладные расходы, увеличение размера журнала

Настройки

- *data_checksums*

ignore_checksum_failure = off

wal_log_hints = off (неявно on при контрольных суммах страниц)

wal_compression = off

Во-вторых, данные могут быть повреждены на носителе, при передаче данных по интерфейсным кабелям и т. п. Часть таких ошибок обрабатывается на аппаратном уровне, но часть — нет.

Чтобы вовремя обнаружить возникшую проблему, журнальные записи всегда снабжаются контрольными суммами.

Страницы данных также можно защитить контрольными суммами при инициализации кластера. В производственной среде это надо делать обязательно, несмотря на накладные расходы на вычисление сумм и их контроль. Иначе можно получить ситуацию, когда возникший сбой не будет вовремя обнаружен.

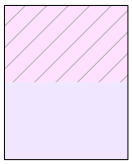
Проверить, включены ли контрольные суммы, можно с помощью параметра data_checksums (только для чтения; включение контрольных сумм «на лету», возможно, появится в PostgreSQL 12). Параметр *ignore_checksum_failure* позволяет не прерывать транзакцию, прочитавшую сбойную страницу, но обычно его не следует включать.

При включенных контрольных суммах в журнал всегда попадает такая «несущественная» информация, как биты подсказок (рассмотрены в модуле «Многоверсионность»), поскольку изменение любого бита приводит и к изменению контрольной суммы. При выключенных контрольных суммах за запись в журнал битов подсказок отвечает параметр *wal_log_hints*.

Изменения битов подсказок всегда журналируется в виде полного образа страницы (FPI, full page image), что сильно увеличивает размер журнала. Но можно включить сжатие полных образов с помощью параметра *wal_compression* (появился в версии 9.5).

Неатомарность записи

Образ страницы



при сбое страница может быть записана не полностью

в журнал записывается образ страницы
при первом ее изменении после контрольной точки

при восстановлении журнальные записи
применяются к записанному образу

Настройки

full_page_writes = on

wal_compression = off

увеличивает размер журнала

В-третьих, есть проблема атомарности записи.

Страница данных занимает 8 КБ (или больше: 16 КБ, 32 КБ), а на низком уровне запись происходит блоками, которые обычно имеют меньший размер (512 байт, 4 КБ, хотя бывают и другие размеры). Поэтому при сбое питания страница данных может записаться частично.

Понятно, что при восстановлении бессмысленно применять к такой странице обычные журнальные записи.

Для защиты PostgreSQL позволяет записывать в журнал образ всей страницы при первом ее изменении после контрольной точки (такой же образ записывается и при изменении битов подсказок) — этим управляет параметр *full_page_writes*. Отключать его имеет смысл, только если используемая файловая система и аппаратура сами по себе гарантируют атомарность записи.

Если при восстановлении мы встречаем в журнале образ страницы, мы безусловно записываем его на диск (к нему больше доверия, т. к. он, как и всякая журнальная запись, защищен контрольной суммой). И уже к нему применяем обычные журнальные записи.

Хотя PostgreSQL исключает из полного образа страницы незанятое место, все же размер журнальных записей увеличивается. Как уже говорилось, размер можно уменьшить за счет сжатия полных образов (параметр *wal_compression*).

Производительность

Характер нагрузки

Синхронная запись

Асинхронная запись

Характер нагрузки

Постоянный поток записи

- последовательная запись, отсутствие случайного доступа
- характер нагрузки отличается от остальной системы
- при высокой нагрузке — размещение на отдельных физических дисках (символьная ссылка из $PGDATA/pg_wal)

Редкое чтение

- при восстановлении
- при работе процессов walsender, если реплика не успевает быстро получать записи

При обычной работе происходит постоянная и последовательная запись журнальных файлов. Поскольку отсутствует случайный доступ, с такой нагрузкой отлично справляются и обычные диски HDD.

Но такой характер нагрузки существенно отличается от того, как происходит доступ к файлам данных. Поэтому обычно выгодно размещать журнал на отдельном физическом диске (дисках), примонтированных к ФС сервера; вместо каталога $PGDATA/pg_wal нужно создать символьную ссылку на соответствующий каталог.

Однако есть ситуация, при которой журнальные файлы необходимо читать (кроме понятного случая восстановления после сбоя). Она возникает, если используется потоковая репликация, и реплика не успевает получать журнальные записи, пока они еще находятся в буферах оперативной памяти основного сервера. Тогда процессу walsender приходится читать нужные данные с диска. Взаимодействие с репликой подробно рассматривается в курсе DBA3 в модуле «Репликация».

Синхронная запись

Алгоритм

при фиксации изменений сбрасывает накопившиеся записи, включая запись о фиксации

ждет *commit_delay*, если активно не менее *commit_siblings* транзакций

Свойства

гарантируется долговечность

увеличивается время отклика

Настройки

synchronous_commit = on
commit_delay = 0
commit_siblings = 5

включать при большом потоке коротких транзакций

Запись журнала происходит в одном из двух режимов: синхронном (когда при фиксации транзакции продолжение работы невозможно до тех пор, пока все журнальные записи о транзакции не окажутся на диске) и асинхронном (когда журнал записывается частями в фоне).

Синхронный режим включается параметром *synchronous_commit.*

Поскольку синхронизация выполняется долго, выгодно выполнять ее как можно реже. Для этого процесс, записывающий журнал, делает паузу, определяемую параметром *commit_delay,* но только в том случае, если имеется не менее *commit_siblings* активных транзакций — расчет на то, что за время ожидания часть транзакций успеют завершиться и можно будет синхронизировать их записи за один раз.

Затем процесс выполняет сброс журнала на диск до необходимого LSN (или несколько больше, если за время ожидания добавились новые записи).

При синхронной записи гарантируется долговечность — если транзакция зафиксирована, то все ее журнальные записи уже на диске и не будут потеряны. Обратная сторона состоит в том, что синхронная запись увеличивает время отклика (команда COMMIT не возвращает управление до окончания синхронизации) и уменьшает производительность системы.

Асинхронная запись

Алгоритм

циклы записи через *wal_writer_delay*

записывает только целиком заполненные страницы;
но если новых полных страниц нет, то записывает последнюю до конца

Свойства

гарантируется согласованность, но не долговечность

зафиксированные изменения могут пропасть (3 × *wal_writer_delay*)

Настройки

synchronous_commit

можно изменять на уровне транзакции

wal_writer_delay = 200ms

При асинхронной записи сброс выполняет процесс wal writer, чередуя циклы работы с ожиданием *wal_writer_delay*. В каждом цикле:

- если с прошлого раза в буферах была целиком заполнена одна или несколько страниц, сбрасываются только такие, полностью заполненные, страницы (или часть таких страниц; вспомним, что журнальный кэш представляет собой кольцевой буфер — записывается только непрерывная последовательность страниц). При большом потоке изменений это позволяет не синхронизировать одну и ту же страницу несколько раз.

- если же заполненные страницы не появились, записывается текущая (не полностью заполненная) страница журнала.

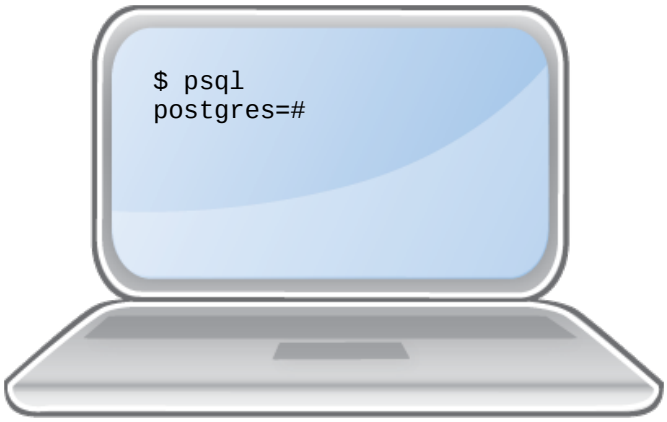
Асинхронная запись эффективнее синхронной — фиксация изменений не ждет записи. Однако надежность уменьшается: зафиксированные данные могут пропасть в случае сбоя, если между фиксацией и сбоем прошло менее 3 × *wal_writer_delay* времени.

Параметр *synchronous_commit* можно устанавливать в рамках транзакций. Это позволяет увеличивать производительность, жертвуя надежностью только части транзакций.

https://postgrespro.ru/docs/postgrespro/10/wal-async-commit

В реальности оба режима работают совместно. Журнальные записи долгой транзакции будут записываться асинхронно (чтобы освободить буферы WAL). А если при сбросе страницы окажется, что соответствующая журнальная запись еще не на диске, она тут же будет сброшена в синхронном режиме.

Демонстрация



Итоги

Журнал позволяет не только восстановить согласованность после сбоев, но может использоваться и для других задач

Надежность записи требует настройки не только СУБД, но и на уровне ОС, ФС и аппаратуры

Настройка позволяет достичь баланса между долговечностью и производительностью

Практика

1. Изучите, как влияет на размер журнальных записей значение параметра *full_page_writes*.

   Для этого повторите простой тест pgbench, показанный в демонстрации, с разными настройками журнала. Перед запуском каждого теста выполняйте контрольную точку.

   Объясните полученный результат.

2. Во сколько раз уменьшается размер журнальных записей при включении параметра *wal_compression*?

1. Обратите внимание на значение параметра *data_checksums*.

Для детального анализа может пригодиться утилита pg_waldump с ключом --stats, показывающим статистическую информацию о составе журнальных записей.