Postgres PROFESSIONAL

Контрольная точка

Темы

Процесс контрольной точки (checkpointer)

Управляющий файл

Размер журнала

Восстановление после сбоя

Настройка

Способы записи буферов

Серверный процесс

- записывает полученный буфер, если он оказался грязным
- плохо, так как замедляется чтение

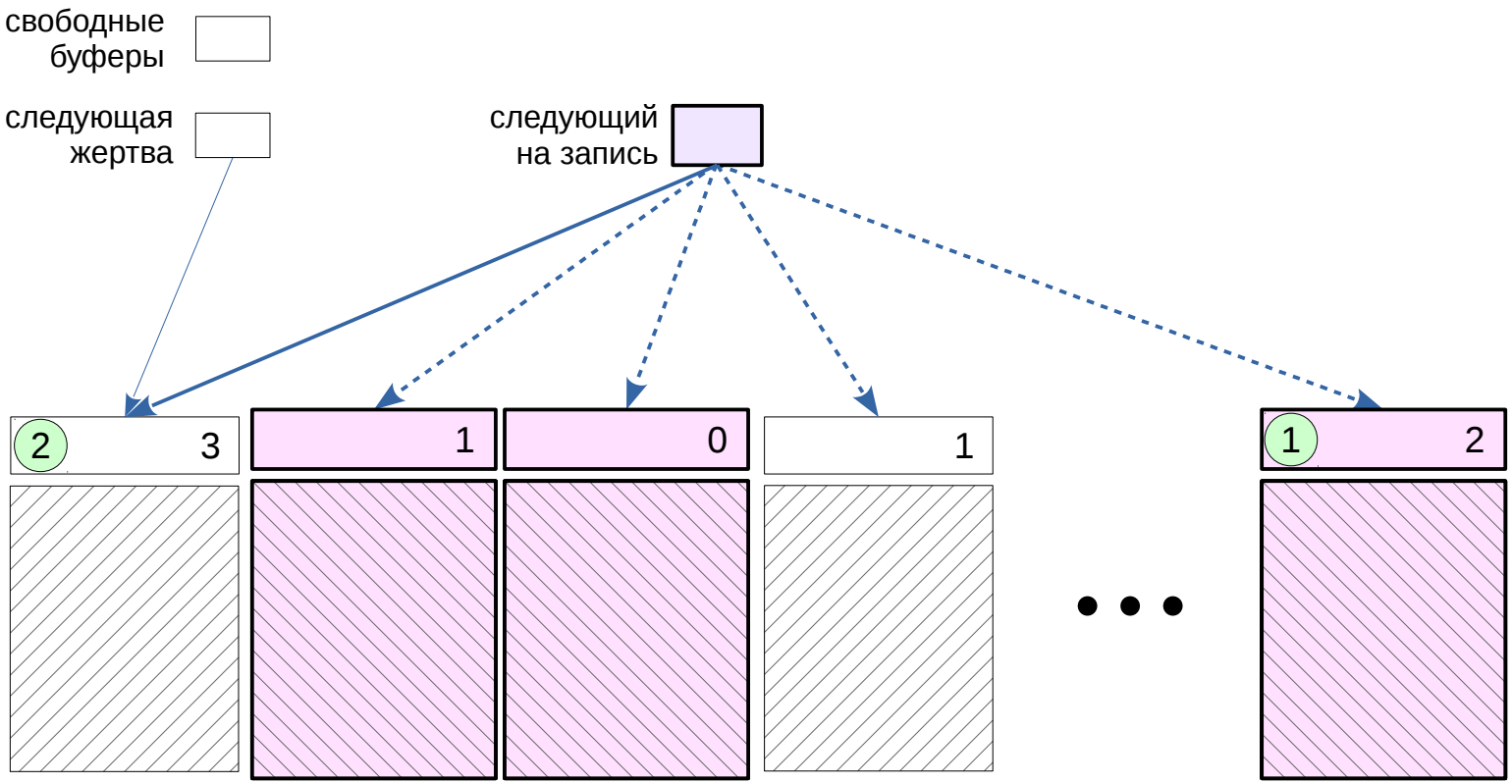
Процесс фоновой записи (writer)

- записывает грязные буферы, которые скоро будут вытеснены
- разгружает серверные процессы

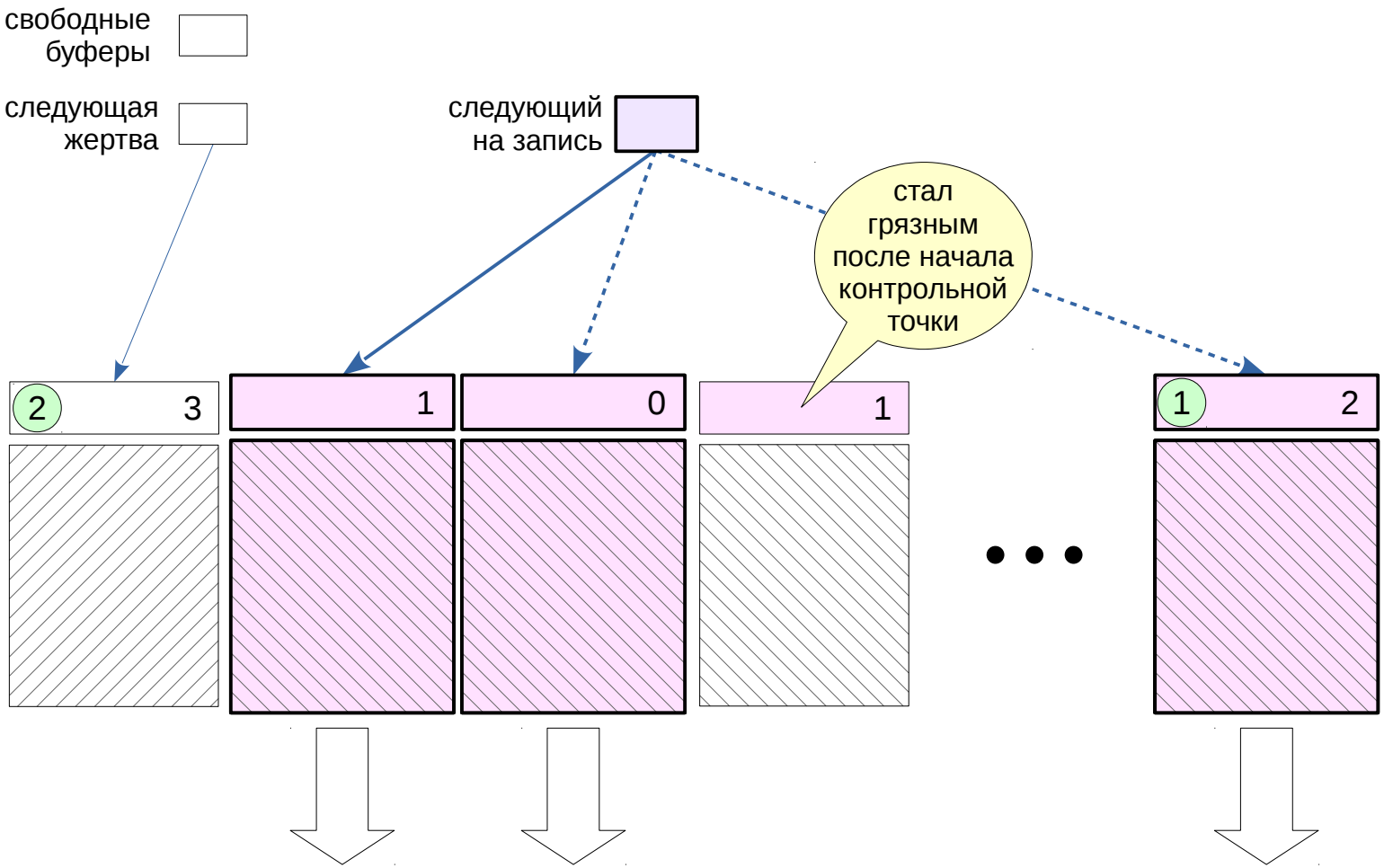
Процесс контрольной точки (checkpointer)

- периодически записывает все грязные буферы
- с контрольной точки начинается восстановление при сбое

Процесс контрольной точки



Процесс контрольной точки



Процесс контрольной точки

Алгоритм

при приближении размера журнала к *max_wal_size*
или после *checkpoint_timeout*:

1. пометить все грязные буферы
2. записать помеченные буферы за *checkpoint_completion_target* от времени между контрольными точками (примерно)
3. записать контрольную точку в файл `pg_control`

Настройки

max_wal_size	= 1GB	(9.5)
checkpoint_segments	= 3	(9.4)
checkpoint_timeout	= 5min	
checkpoint_completion_target	= 0.5	

Размер журнала

Поддерживается в определенных границах

минимальный — данные для восстановления и резерв места

максимальный — зависит от частоты контрольных точек

При освобождении сегмента

(не нужен для восстановления, репликации, *wal_keep_segments*)

сегмент либо удаляется,
либо переименовывается и используется заново

Настройки

min_wal_size	= 80MB	(9.5)
max_wal_size	= 1GB	(9.5)
checkpoint_segments	= 3	(9.4)
wal_keep_segments	= 0	

Настройка

Контрольные точки

частые

избыточный ввод-вывод

редкие

больше размер журнала

больше время восстановления

Мониторинг

pg_stat_bgwriter

log_checkpoints, *checkpoint_warning*

Порядок настройки

процесс контрольной точки

процесс фоновой записи — в помощь

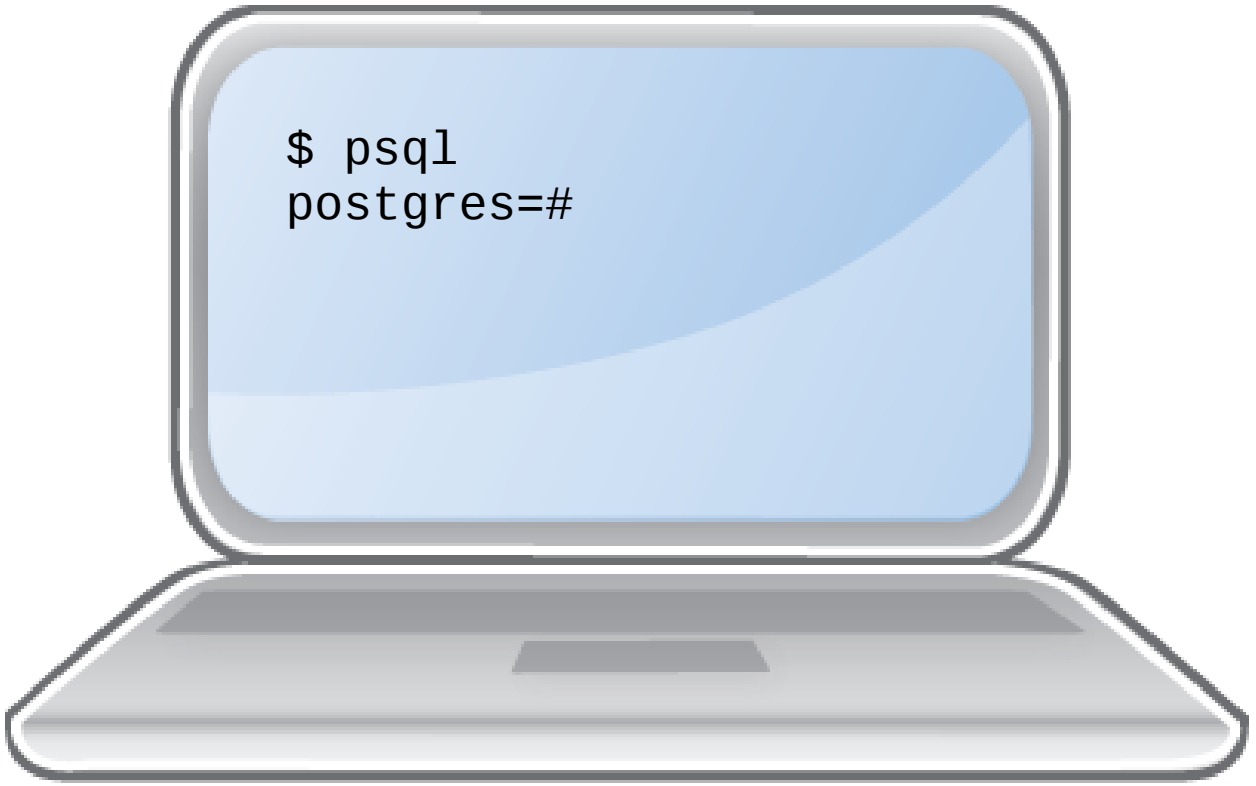
Восстановление

Алгоритм

при старте сервера после сбоя
(состояние кластера в `pg_control` отличается от «shut down»):

1. прочитать LSN_0 последней контрольной точки из `pg_control`
2. для каждой записи журнала, начиная с LSN_0:
   - 3.1. определить страницу, к которой относится запись
   - 3.2. применить запись, если LSN страницы меньше, чем LSN записи
3. перезаписать нежурналируемые таблицы init-файлами
4. выполнить контрольную точку

Демонстрация



Итоги

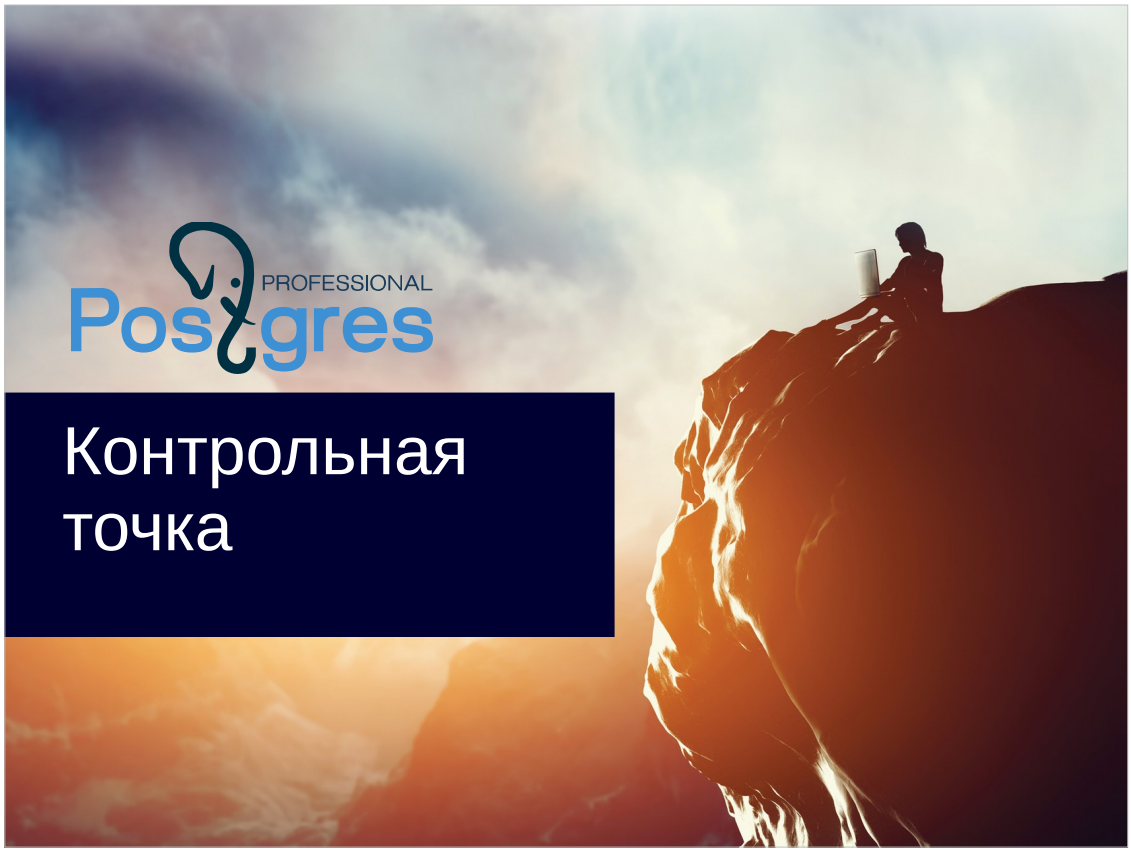
Процесс контрольной точки записывает на диск все грязные буферы

Работает вместе с процессом фоновой записи

Восстановление начинается с контрольной точки, более ранние журнальные записи не требуются

Практика

1. Создайте базу DB9.
2. Настройте выполнение контрольной точки раз в 30 секунд, включите вывод контрольных точек в журнал сообщений, отключите процесс фоновой записи.
3. Сбросьте статистику.
4. В течение нескольких минут изменяйте данные в базе DB9.
5. Какой объем журнала был сгенерирован за это время?
6. Учитывая значение параметра *max_wal_size*, достаточно ли выполнения контрольной точки по расписанию? Проверьте по статистике, сколько раз выполнялась контрольная точка.
7. Проверьте журнал сообщений. Как объяснить время записи контрольной точки?



Авторы: Егор Рогов, Павел Лузанов

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

Темы

Процесс контрольной точки (checkpointer)

Управляющий файл

Размер журнала

Восстановление после сбоя

Настройка

Способы записи буферов

Серверный процесс

- записывает полученный буфер, если он оказался грязным
- плохо, так как замедляется чтение

Процесс фоновой записи (writer)

- записывает грязные буферы, которые скоро будут вытеснены
- разгружает серверные процессы

Процесс контрольной точки (checkpointer)

- периодически записывает все грязные буферы
- с контрольной точки начинается восстановление при сбое

Для измененной страницы есть три способа попасть на диск.

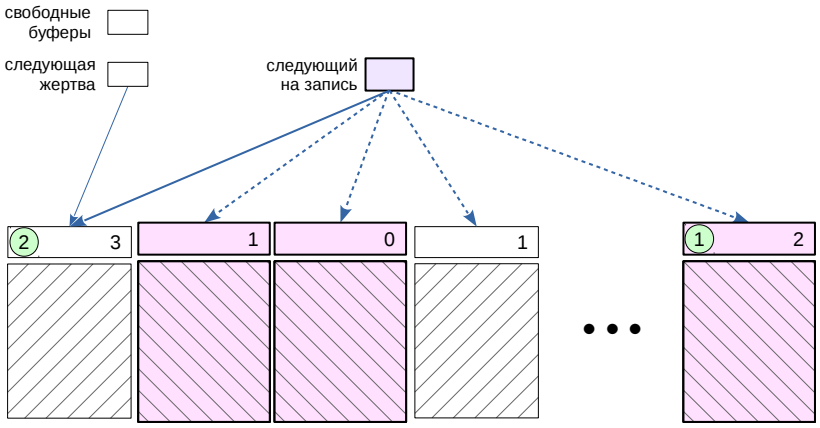
Первые два были рассмотрен в теме «Буферный кэш».

Когда серверному процессу требуется буфер, он выбирается среди существующих механизмом вытеснения. Если буфер оказывается грязным, серверному процессу приходится записать находящуюся в нем страницу, прежде чем он сможет занять буфер своей страницей.

Поскольку это неэффективно — чтение будет происходить медленее, чем могло бы — существует фоновый процесс записи. Он находит буферы, которые с большой вероятностью будут вытеснены в ближайшее время, и записывает страницы.

Наконец, третий способ — процесс контрольной точки. Он периодически записывает на диск вообще все грязные буферы. Смысл этой операции состоит в том, чтобы ограничить размер журналов упреждающей записи, необходимых для восстановления после сбоя. Так как выполнение контрольной точки гарантирует, что все измененные страницы из буферного кэша записаны на диск, то для восстановления достаточно иметь журналы с момента начала контрольной точки.

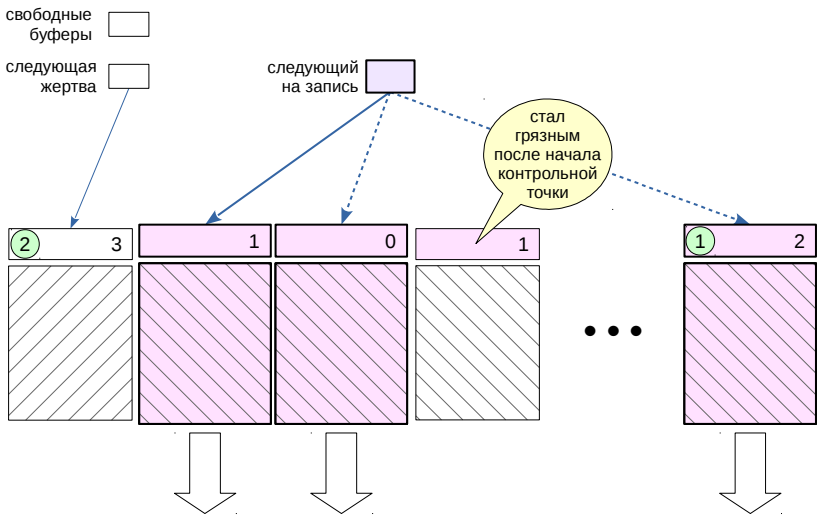
Процесс контрольной точки



Процесс контрольной точки должен записать на диск все грязные буферы, которые имели место на начало контрольной точки.

Для этого сначала он выставляет на грязных буферах признак того, что их необходимо записать.

Процесс контрольной точки



Затем он проходит по всем помеченным буферам и записывает их на диск. Порядок обхода буферов совпадает с тем порядком, которого придерживается фоновый процесс записи, чтобы в первую очередь записать те буферы, которые могут быть вытеснены. Однако процесс контрольной точки записывает все грязные буферы, не смотря на число обращений и число ссылок.

Помеченные буферы могут также быть записаны серверными процессами или фоновым процессом записи — смотря кто доберется до буфера первым. В любом случае при записи снимается признак, так что буфер не будет записан несколько раз.

Если в процессе записи какие-то страницы изменяются, они не рассматриваются процессом контрольной точки, так как на момент начала они не были грязными.

Процесс контрольной точки

Алгоритм

при приближении размера журнала к *max_wal_size*
или после *checkpoint_timeout*:

1. пометить все грязные буферы
2. записать помеченные буферы за *checkpoint_completion_target* от времени между контрольными точками (примерно)
3. записать контрольную точку в файл `pg_control`

Настройки

max_wal_size	= 1GB	(9.5)
checkpoint_segments	= 3	(9.4)
checkpoint_timeout	= 5min	
checkpoint_completion_target	= 0.5	

Процесс контрольной точки начинается в одном из двух случаев:

- 9.5: размер журнала (все сегменты в pg_xlog) приближается к *max_wal_size,*

- 9.4: с прошлой контрольной точки заполнено *checkpoint_segments* сегментов,

- прошло *checkpoint_timeout* с момента прошлой контрольной точки.

Исключение составляет случай, когда в системе не происходит никакой активности — тогда контрольная точка не выполняется.

Контрольную точку можно выполнить и вручную командой checkpoint.

Чтобы распределить нагрузку на подсистему ввода-вывода, контрольная точка может быть растянута во времени с помощью параметра *checkpoint_completion_time.* Он определяет продолжительность записи от расстояния между двумя контрольными точками.

http://www.postgresql.org/docs/current/static/wal-configuration.html

http://www.postgresql.org/docs/current/static/runtime-config-wal.html

После окончания записи страниц информация о пройденной контрольной точке записывается в специальный управляющий файл pg_control (в частности, LSN записи об этой контрольной точке). Содержимое файла удобно просматривать утилитой pg_controldata.

Размер журнала

Поддерживается в определенных границах

минимальный — данные для восстановления и резерв места

максимальный — зависит от частоты контрольных точек

При освобождении сегмента

(не нужен для восстановления, репликации, *wal_keep_segments*)

сегмент либо удаляется,
либо переименовывается и используется заново

Настройки

min_wal_size	= 80MB	(9.5)
max_wal_size	= 1GB	(9.5)
checkpoint_segments	= 3	(9.4)
wal_keep_segments	= 0	

Постгрес пытается удержать размер журнала (иными словами, количество сегментов в $PGDATA/pg_xlog) в определенных границах.

Всегда поддерживается некоторое минимальное число сегментов: еще необходимые для восстановления после сбоя плюс заранее зарезервированные сегменты для новых записей.

Если сегмент освобождается (то есть он больше не нужен для восстановления, его не держит репликационный слот (рассматривается в теме «Потоковая репликация») или параметр *wal_keep_segments*), и при этом число сегментов не превышает минимальное, сегмент переименовывается и используется заново.

Если же сегментов больше, то освободившийся сегмент удаляется.

Таким образом, максимальный размер журнала в основном ограничен частотой выполнения контрольных точек.

В версии 9.5 для определения границ используются параметры *min_wal_size* и *max_wal_size*. Последний параметр определяет частоту контрольных точек.

В версии 9.4 используется один параметр *checkpoint_segments,* определяющий частоту контрольных точек, а границы вычисляются исходя из него.

http://www.postgresql.org/docs/9.4/static/wal-configuration.html

Настройка

Контрольные точки

частые	*редкие*
избыточный ввод-вывод	больше размер журнала больше время восстановления

Мониторинг

pg_stat_bgwriter

log_checkpoints, checkpoint_warning

Порядок настройки

процесс контрольной точки

процесс фоновой записи — в помощь

В правильно настроенной системе записью грязных буферов занимаются процессы контрольной точки и фоновой записи. Серверные процессы как правило не должны записывать буферы самостоятельно.

Настройка контрольной точки — это поиск баланса между излишним вводом-выводом (при частых контрольных точках) и большим объемом журнала и увеличением времени восстановления при сбоях (при редких контрольных точках).

Обычно параметр checkpoint_timeout настраивается так, чтобы в большинстве случаев контрольные точки происходили по расписанию. Зная объем журналов, генерируемых за это время (при заданной нагрузке), можно рассчитать необходимый размер журнальных файлов. Параметр max_wal_size устанавливается выше этого значения и ограничивает размер журналов сверху.

Для мониторинга следует использовать представление pg_stat_bgwriter: поля checkpoints_timed и checkpoints_req покажут количество выполненных контрольных точек по расписанию и по требованию. Также полезны параметры log_checkpoints и checkpoint_warning.

Процесс фоновой записи настраивается так, чтобы (совместно с контрольной точкой) грязные буферы успевали записываться до того, как потребуются серверным процессам. Представление pg_stat_bgwriter покажет количество страниц, записанных разными процессами.

Чтобы избежать пиков нагрузки на подсистему ввода-вывода, имеет смысл растягивать контрольную точку (вплоть до 90% времени между точками) и ограничивать аппетиты процесса фоновой записи.

Восстановление

Алгоритм

при старте сервера после сбоя
(состояние кластера в `pg_control` отличается от «shut down»):

1. прочитать LSN_0 последней контрольной точки из `pg_control`
2. для каждой записи журнала, начиная с LSN_0:
   3.1. определить страницу, к которой относится запись
   3.2. применить запись, если LSN страницы меньше, чем LSN записи
3. перезаписать нежурналируемые таблицы init-файлами
4. выполнить контрольную точку

Если в работе сервера произошел сбой, то при последующем запуске процесс startup (запускаемый postmaster-ом в самом начале работы) обнаружит это, посмотрев в файл pg_control и увидев статус, отличный от «shut down». Тогда будет выполнено восстановление.

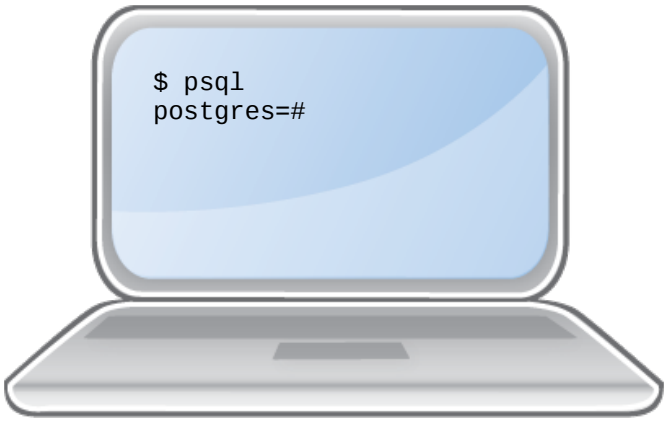
Сначала процесс прочитает из того же pg_control LSN записи о последней контрольной точке. Дальше он будет читать журнал от этой позиции, последовательно применяя записи к страницам, если в этом есть необходимость (что можно проверить, сравнив LSN страницы на диске с LSN журнальной записи). Аналогично записи применяются и к файлам: например, если запись говорит о том, что файл долен быть создан, а его нет — файл создается.

В конце процесса все нежурналируемые таблицы перезаписываются с помощью образов в init-файлах.

На этом процесс startup завершает работу и управление передается процессу postmaster.

Как только postmaster запустит процесс checkpointer, будет выполнена контрольная точка, чтобы зафиксировать восстановленное состояние.

Демонстрация



Итоги

Процесс контрольной точки записывает на диск все грязные буферы

Работает вместе с процессом фоновой записи

Восстановление начинается с контрольной точки, более ранние журнальные записи не требуются

Практика

1. Создайте базу DB9.
2. Настройте выполнение контрольной точки раз в 30 секунд, включите вывод контрольных точек в журнал сообщений, отключите процесс фоновой записи.
3. Сбросьте статистику.
4. В течение нескольких минут изменяйте данные в базе DB9.
5. Какой объем журнала был сгенерирован за это время?
6. Учитывая значение параметра *max_wal_size*, достаточно ли выполнения контрольной точки по расписанию? Проверьте по статистике, сколько раз выполнялась контрольная точка.
7. Проверьте журнал сообщений. Как объяснить время записи контрольной точки?

Нагрузку удобно имитировать с помощью расширения pgbench.