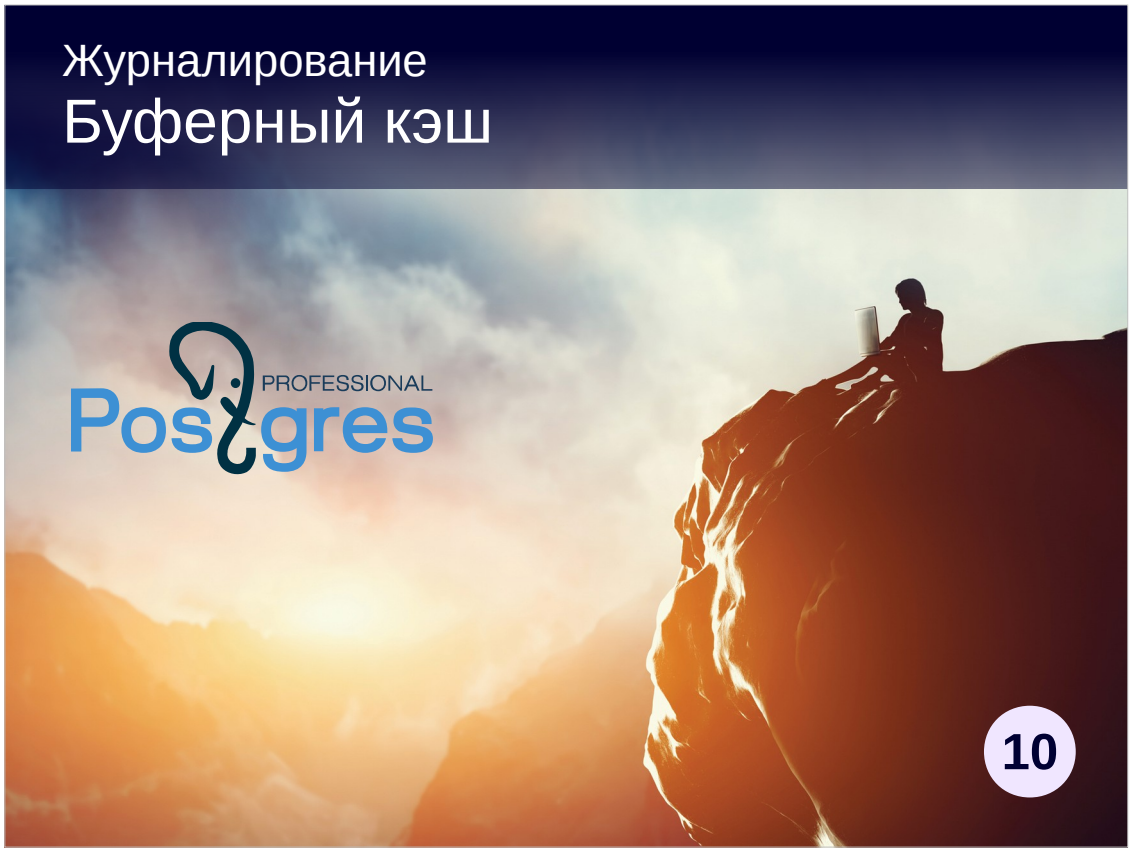


Авторы: Егор Рогов, Павел Лузанов

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

Темы

Устройство и использование буферного кэша

Механизм вытеснения страниц

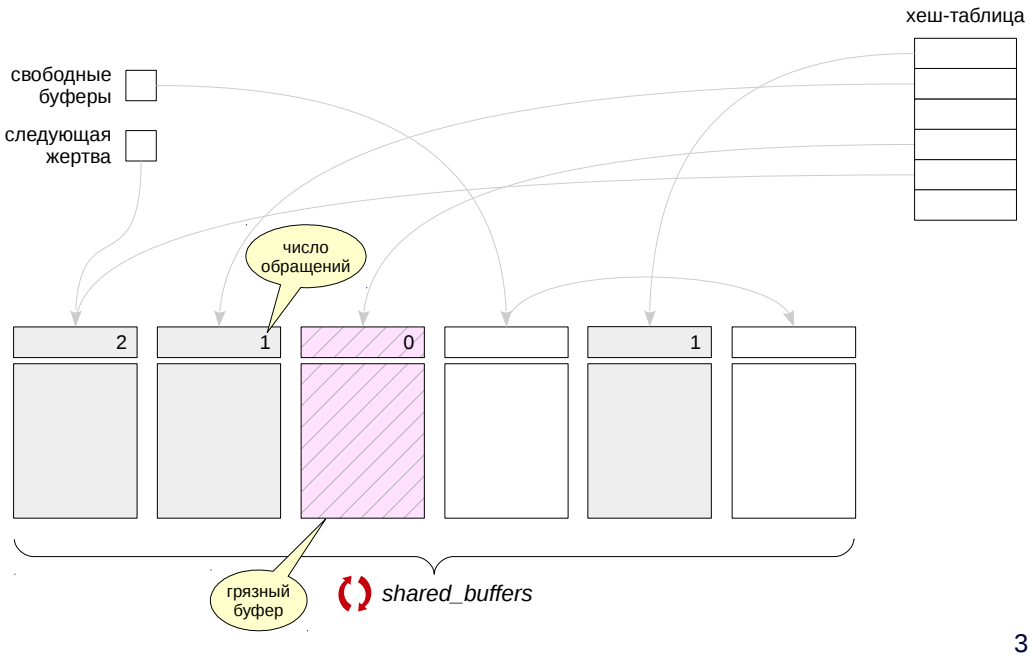
Массовое вытеснение и буферные кольца

Настройка размера кэша

Локальный кэш для временных таблиц

Прогрев кэша

Буферный кэш



Задача буферного кэша — сглаживать разницу в производительности двух типов памяти: оперативной (быстрая, но мало) и дисковой (медленная, но много). Чтобы работать с данными — читать или изменять, — процессы читают страницы в буферный кэш. Пока страница находится в кэше, мы экономим на обращениях к диску.

Буферный кэш располагается в общей памяти сервера и представляет собой массив буферов. Каждый буфер состоит из места под одну страницу данных и заголовка. Заголовок содержит, в числе прочего:

- расположение страницы на диске (файл и номер страницы в нем);

- число обращений к буферу (счетчик увеличивается каждый раз, когда процесс читает или изменяет буфер);

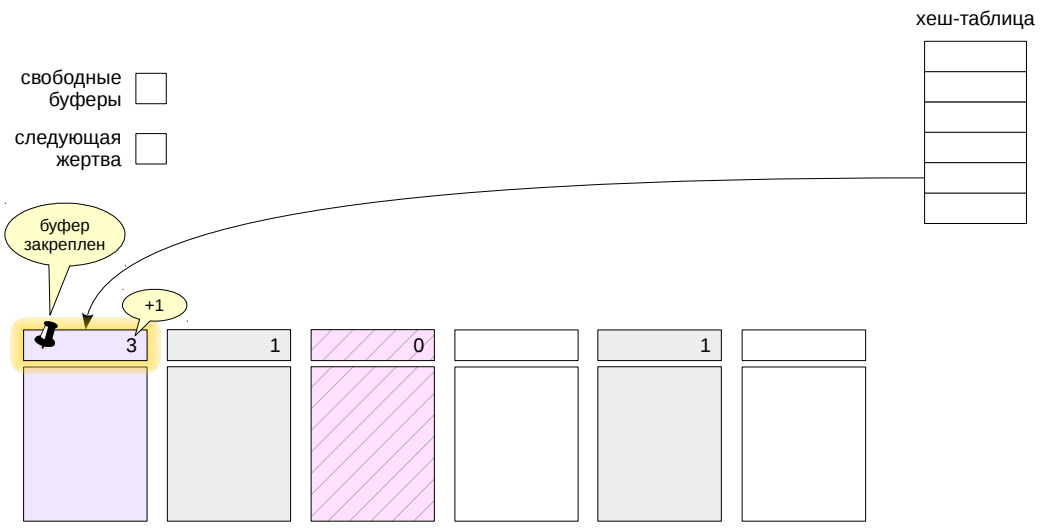
- признак того, что данные на странице изменились и рано или поздно должны быть записаны на диск (такой буфер называют грязным);

Размер буферного кэша задается параметром *shared_buffers*. Его изменение требует перезапуска сервера.

Изначально кэш содержит пустые буферы, и все они связаны в список свободных буферов. Смысл указателя на «следующую жертву» станет ясен чуть позже.

Чтобы быстро находить нужную страницу в кэше, используется хеш-таблица.

Страница в кэше



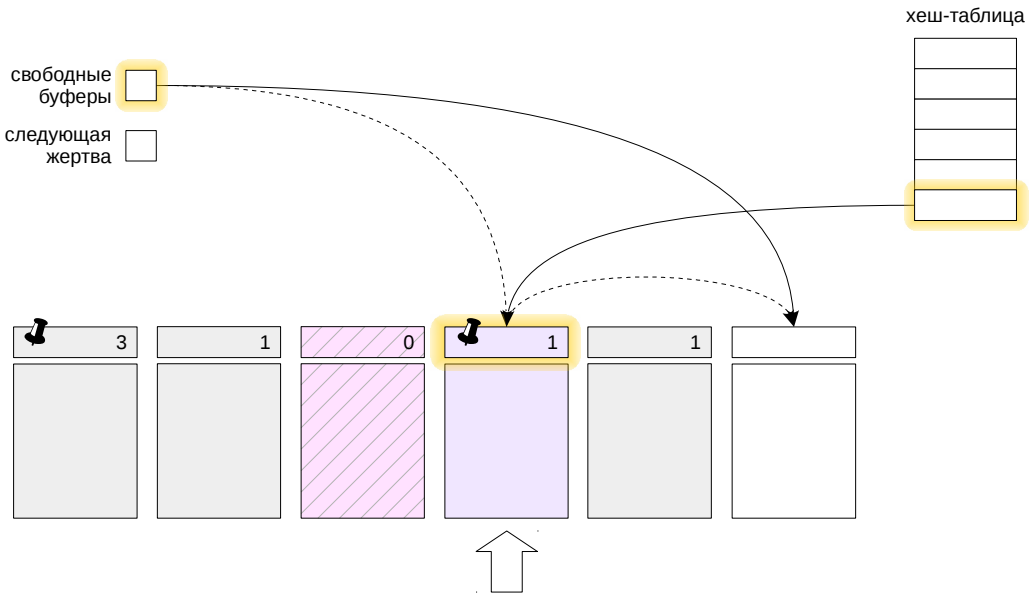
Когда процессу требуется прочитать страницу, он сначала пытается найти ее в буферном кэше с помощью хеш-таблицы.

Ключом хеширования служит файл и номер страницы внутри файла; получив номер буфера, процесс быстро проверяет, действительно ли он содержат нужную страницу. Как и в любой хеш-таблице, здесь возможны коллизии; в таком случае процессу придется проверять несколько страниц.

Если нужная страница найдена в кэше, процесс должен «закрепить» буфер (увеличить счетчик pin count) и увеличить число обращений (счетчик usage count).

Пока буфер закреплен (значение pin count больше нуля), считается, что буфер используется и его содержимое не должно «радикально» измениться. Например, в странице может появиться новая версия строки — это никому не мешает благодаря многоверсионности и правилам видимости. Но в закрепленный буфер не может быть прочитана другая страница.

Чтение в свободный буфер



Может получиться так, что необходимая страница не будет найдена в кэше. В этом случае ее необходимо считать с диска в какой-либо буфер.

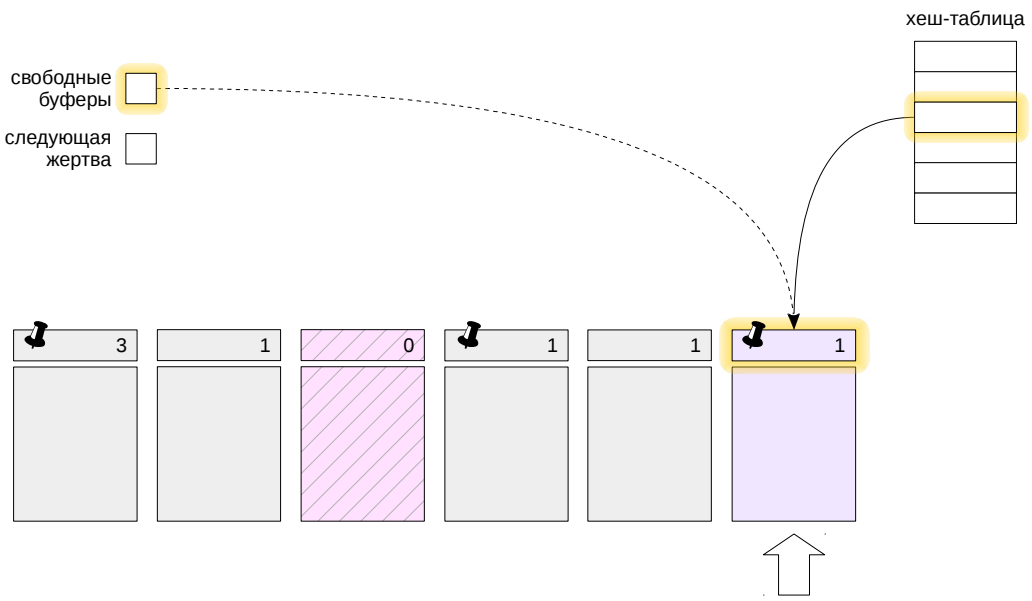
В первую очередь поиск подходящего буфера будет проходить по списку свободных буферов, если он не пуст.

Найденный по указателю буфер закрепляется, в него читается необходимая страница, число обращений устанавливается в единицу.

Указатель на свободный буфер передвигается на следующий по списку буфер, если он есть.

Кроме того, ссылку на загруженную страницу необходимо прописать в хеш-таблицу, чтобы в дальнейшем ее можно было найти.

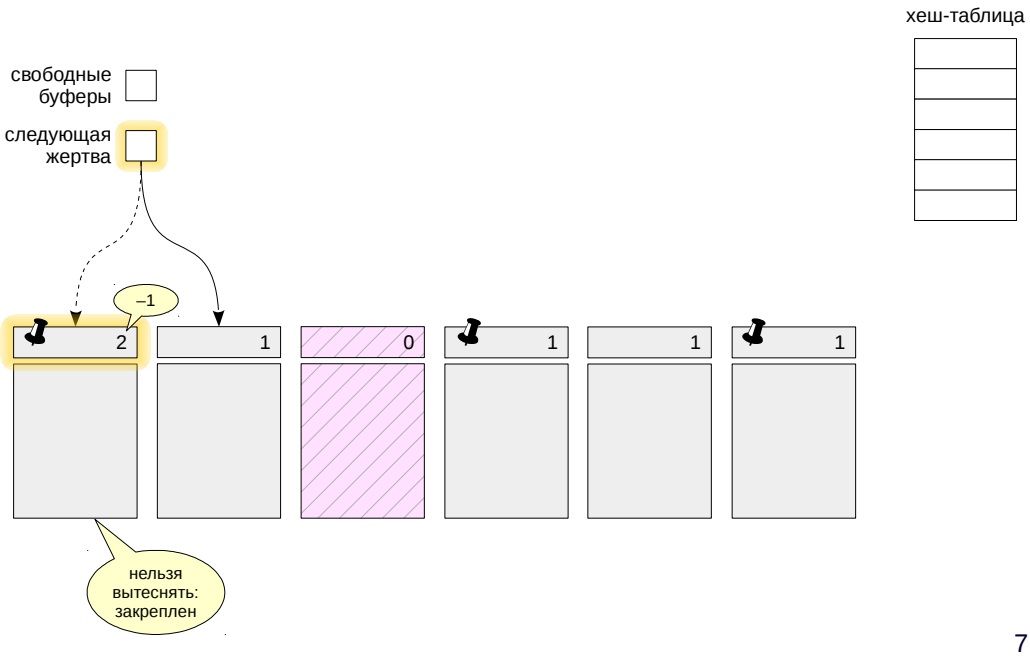
Чтение в свободный буфер



В конце концов будет заполнен и последний свободный буфер.

Теперь ссылка на свободный буфер пуста — в дальнейшем, чтобы прочитать новую страницу в любой буфер, придется из этого буфера вытеснить страницу, которая там находится.

Чтение с вытеснением



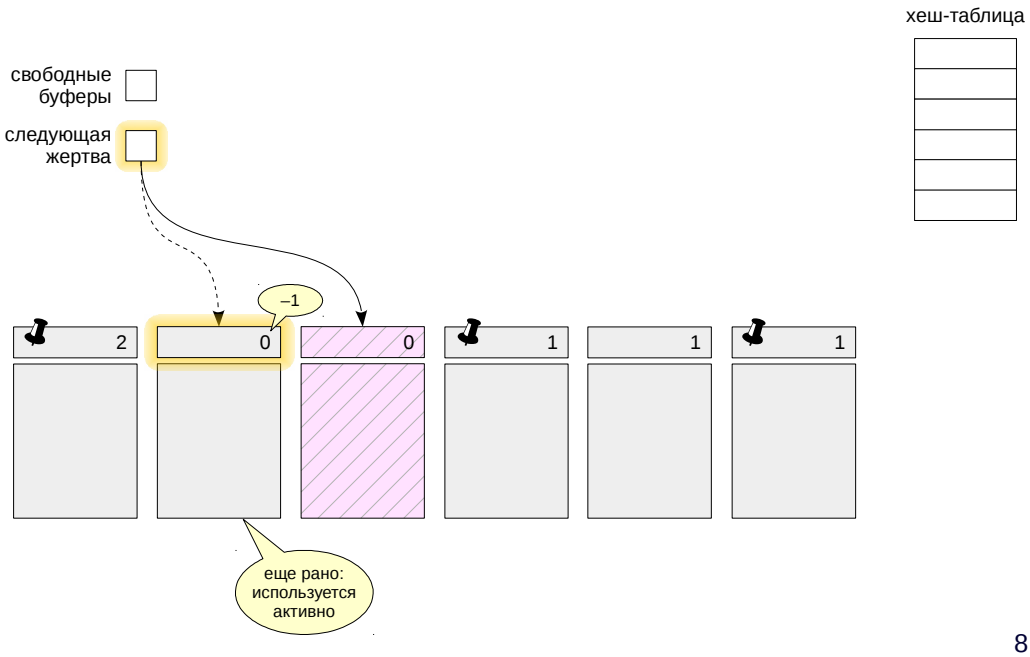
Механизм вытеснения использует указатель на следующую «жертву».

Алгоритм состоит в том, чтобы перебирать по кругу все буферы, уменьшая их счетчики обращений (usage count). Выбирается первый же буфер, у которого значение счетчика равно 0 (и который в принципе может быть занят новой страницей). По-английски этот алгоритм называется clock-sweep.

Чем больше значение счетчика у буфера (то есть чем чаще он используется), тем больше у него шансов задержаться в кэше. Максимальное значение счетчика обращений ограничено числом 5, чтобы избежать «наматывания кругов» при больших значениях.

В нашем примере процесс обращается к буферу по указателю. Буфер закреплен (то есть используется каким-то процессом), и поэтому страница не может быть вытеснена из него. Тем не менее мы уменьшаем значение счетчика обращений на единицу и переходим к следующему буферу.

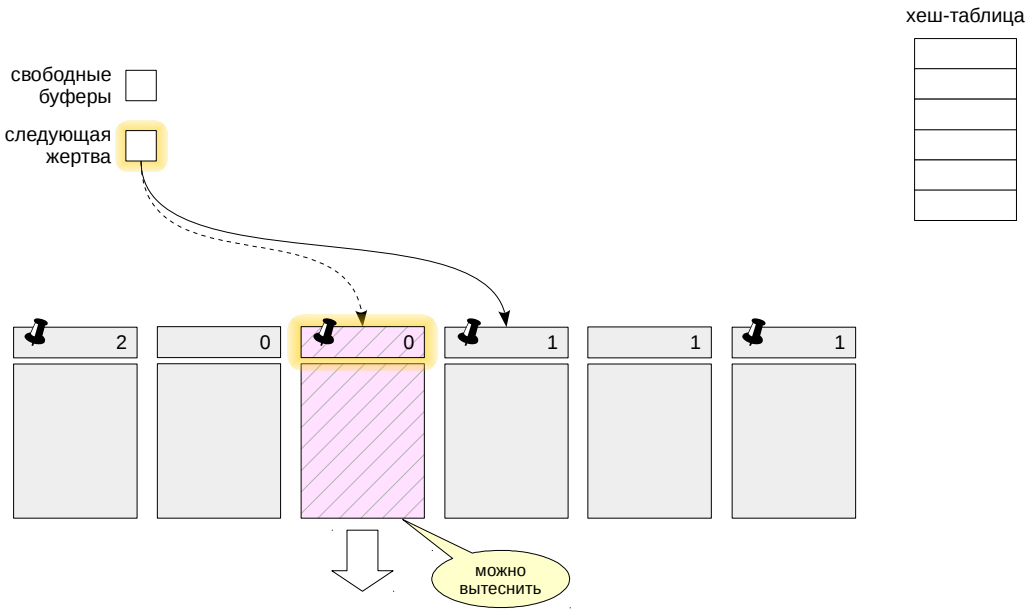
Чтение с вытеснением



Следующий буфер не закреплен, однако его счетчик обращений больше нуля.

Мы уменьшаем счетчик на единицу и даем ему шанс остаться в кэше.

Чтение с вытеснением

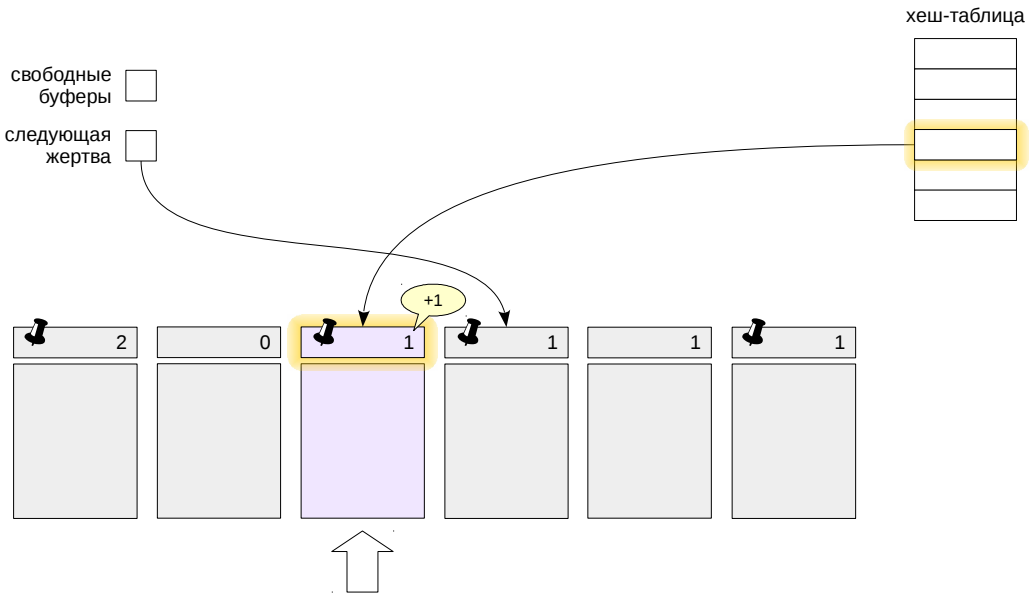


Наконец, мы приходим к незакрепленному буферу с нулевым счетчиком обращений. Страница из этого буфера и будет вытеснена.

Однако в нашем примере найденный буфер оказался грязным — он содержит измененные данные. Поэтому сначала страницу требуется сохранить на диск. Для этого сначала буфер закрепляется (чтобы показать остальным процессам, что он используется), после чего страница записывается на диск.

Это не очень хорошая ситуация: процессу, который собирался прочитать страницу, придется ждать, пока «чужие» данные будут записаны. Этот эффект сглаживается процессами контрольной точки и фоновой записи, которые рассмотрены в теме «Контрольная точка».

Чтение с вытеснением



После того, как страница из грязного буфера записана на диск, в освободившийся буфер ранее рассмотренным образом читается новая страница.

Ссылка на следующую «жертву» уже указывает на следующий буфер, а у только что загруженного есть время нарастить счетчик обращений, пока указатель не обойдет по кругу весь буферный кэш и не вернется вновь.

Массовое вытеснение

Буферное кольцо

часть буферного кэша, выделенная для одной операции

предотвращает вытеснение кэша «одноразовыми» данными

операция	*кол-во страниц*	*грязные буферы*
последовательное чтение	32	исключаются из кольца
очистка (VACUUM)	32	вытесняются на диск
массовая запись (COPY, CTAS)	≤2048	вытесняются на диск

При операциях, выполняющих массовое чтение или запись данных, есть опасность быстрого вытеснения полезных страниц из буферного кэша «одноразовыми» данными.

Чтобы этого не происходило, для таких операций используются так называемые буферные кольца — для каждой операции выделяется небольшая часть буферного кэша. При этом вытеснение действует только в пределах кольца, поэтому остальные данные в буфере не страдают.

Для последовательного чтения таблицы (sequential scan) грязные буферы не вытесняются, а отключаются от кольца и возвращаются в основной кэш — они будут вытеснены «на общих основаниях». Вместо отключенного буфера в кольцо из кэша подключается другой буфер. Такая стратегия рассчитана на то, что данные будут в основном читаться (и это надо делать быстро), а не записываться.

Если в процессе чтения таблицы другому процессу тоже потребуются эти данные, он не начинает читать таблицу сначала, а подключается к уже имеющемуся буферному кольцу. После окончания сканирования он дочитывает «пропущенное» начало таблицы.

Для процесса очистки используется кольцо небольшого размера (32 страницы), так как замедление фоновой задачи не столь критично, как замедление пользовательских процессов.

Наоборот, для массовых операций записи — COPY, CREATE TABLE AS SELECT — кольцо имеет достаточно большой размер (обычно 2048 страниц, но не больше одной восьмой всего буферного кэша).

Настройка размера кэша

Настройка

shared_buffers = 128MB ↑

Буферный кэш должен содержать «активные» данные

- при меньшем размере постоянно вытесняются полезные страницы
- при большем размере бессмысленно растут накладные расходы
- начальное приближение — 0.25 ОЗУ

Нужно учитывать двойное кэширование

- если страницы нет в кэше СУБД, она может оказаться в кэше ОС
- алгоритм вытеснения ОС не учитывает специфики базы данных

Размер кэша устанавливается параметром *shared_buffers*. Значение по умолчанию — 128 МБ — сильно занижено.

Как выбрать подходящее значение? Даже самая большая база имеет ограниченный набор данных, с которыми ведется активная работа. В идеале этот набор должен помещаться в буферный кэш (плюс некоторое место для «одноразовых» данных).

Если размер кэша будет меньше, то активно используемые страницы будут постоянно вытеснять друг друга, создавая избыточный ввод-вывод. Признаком такой ситуации может служить то, что все страницы в кэше имеют большое число обращений (usage count).

Однако при большом размере кэша будут расти накладные расходы на его поддержание.

В качестве первого приближения можно взять 1/4 оперативной памяти, но надо понимать, что оптимальное значение зависит не от размера ОЗУ, а от конкретных данных и нагрузки. (Для Windows до версии PostgreSQL 10 рекомендовалось ставить размер меньше.)

Не следует забывать и о том, что PostgreSQL работает с диском через операционную систему и, таким образом, происходит двойное кэширование: страницы попадают как в буферный кэш, так и в кэш ОС. Таким образом, «непопадание» в буферный кэш не всегда приводит к необходимости реального ввода-вывода. Но стратегия вытеснения ОС не учитывает специфики баз данных: например, полное чтение таблицы не вытесняет буферный кэш, но вытесняет кэш ОС.

https://postgrespro.ru/docs/postgresql/10/runtime-config-resource#RUNTIME-CONFIG-RESOURCE-MEMORY

Временные таблицы

Данные временных таблиц

- видны только одному сеансу — нет смысла использовать общий кэш
- существуют в пределах сеанса — не жалко потерять при сбое

Используется локальный буферный кэш

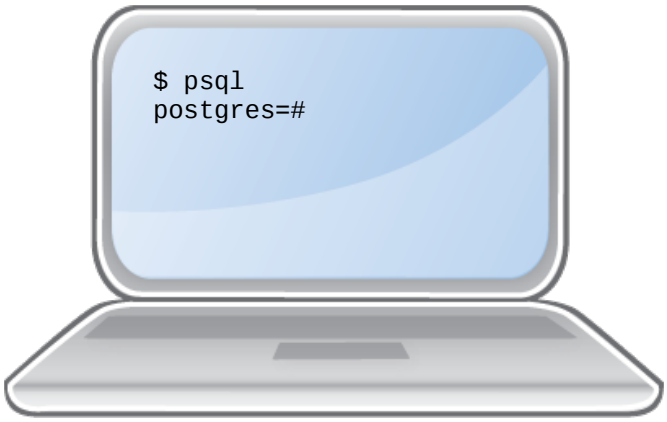
- не требуются блокировки
- память выделяется по необходимости в пределах *temp_buffers*
- обычный алгоритм вытеснения

Исключение из общего правила представляют собой временные таблицы. Поскольку временные данные видны только одному процессу, им нечего делать в общем буферном кэше. Более того, временные данные существуют только в рамках одного сеанса, так что их не нужно защищать от сбоя.

Для временных данных используется облегченный локальный кэш.

Поскольку локальный кэш доступен только одному процессу, для него не требуется блокировки. Память выделяется по мере необходимости (в пределах, заданных параметром *temp_buffers*), ведь временные таблицы используются далеко не во всех сеансах. В локальном кэше используется обычный алгоритм вытеснения.

Демонстрация



Итоги

Вся работа с данными происходит через буферный кэш

Редко используемые страницы вытесняются,
часто используемые — остаются

Буферный кэш сокращает объем ввода-вывода,
но приводит к необходимости журналирования

Практика

1. Создайте таблицу и вставьте в нее некоторое количество строк.
2. Определите, сколько занимает таблица
   а) страниц на диске,
   б) буферов в кэше.
3. Узнайте количество грязных буферов в кэше на текущий момент.
4. Выполните контрольную точку командой CHECKPOINT. Сколько грязных буферов осталось теперь?

Чтобы проверить содержимое буферного кэша, используйте расширение pg_buffercache:
https://postgrespro.ru/docs/postgresql/10/pgbuffercache.html