Postgres PROFESSIONAL

Буферный кэш

Темы

Блокировки в памяти

Устройство и использование буферного кэша

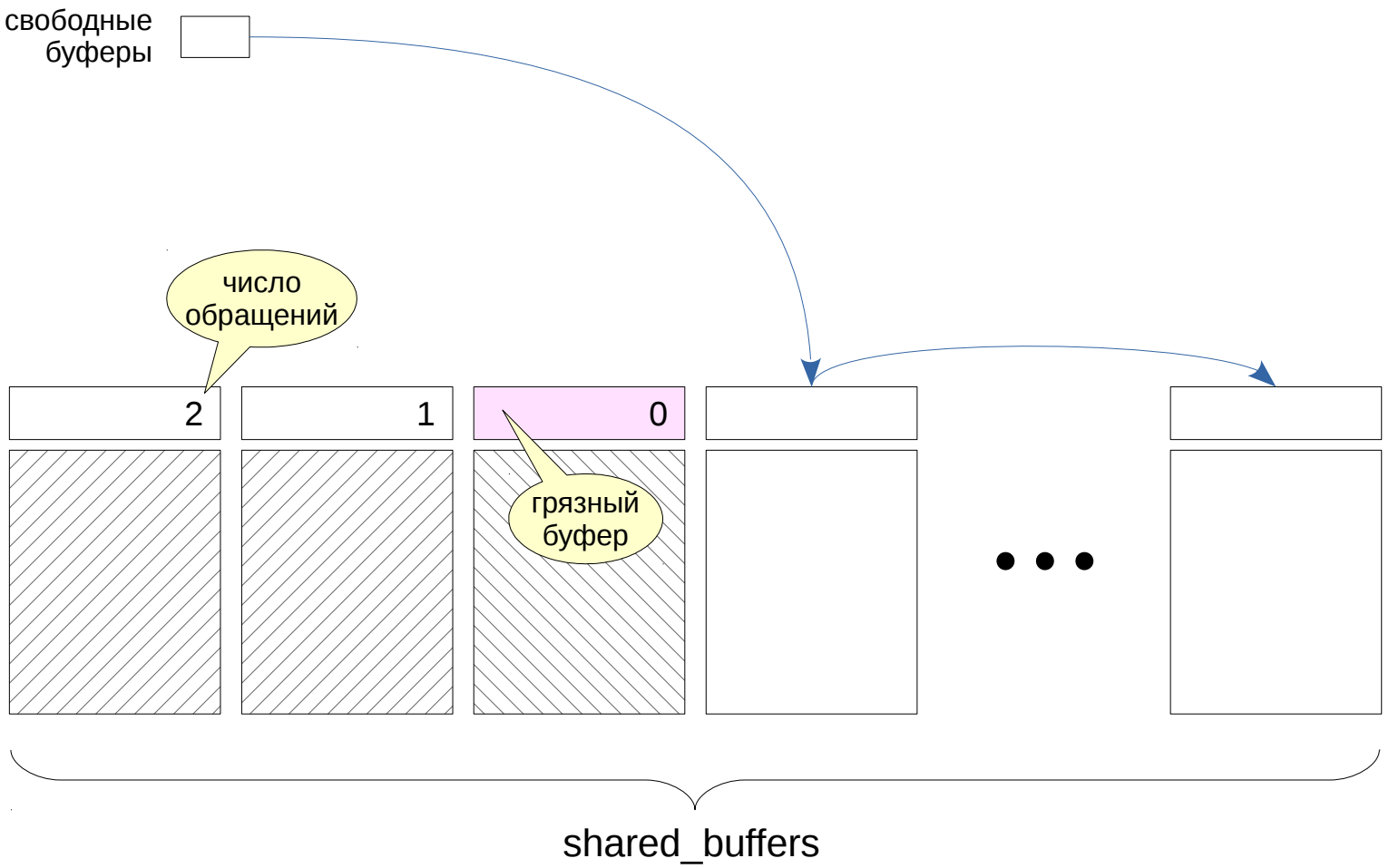
Механизм вытеснения

Запись грязных буферов

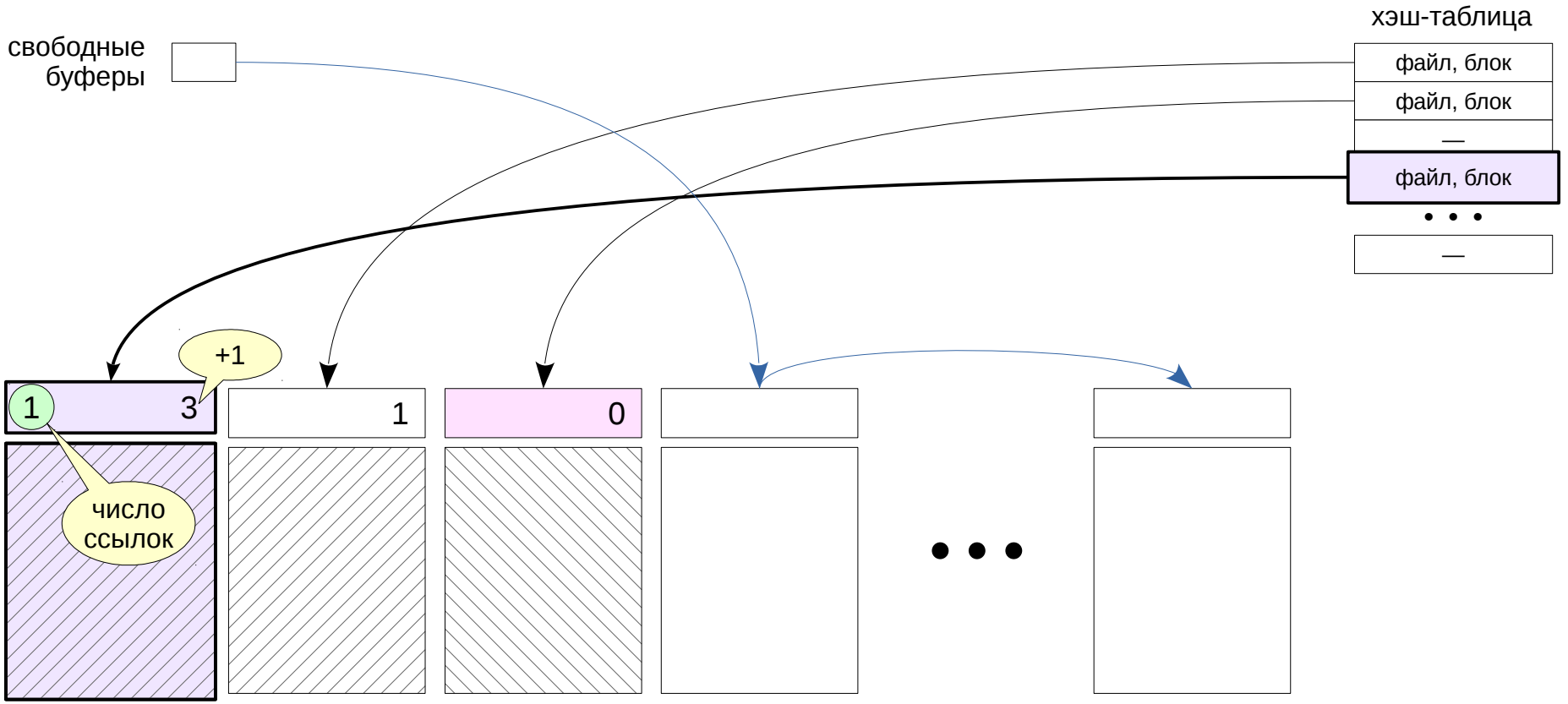
Процесс фоновой записи (writer)

Настройка размера кэша

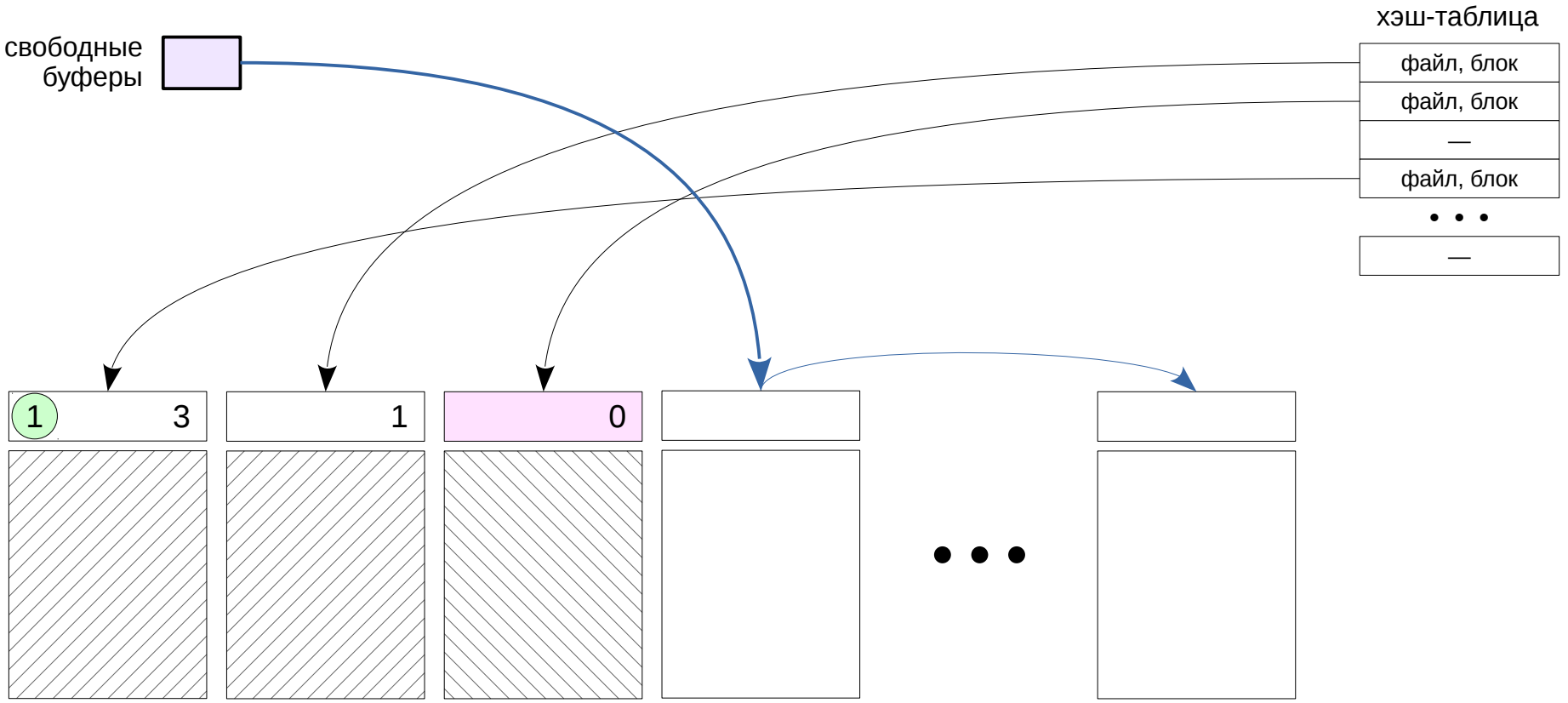
Чтение страницы из кэша



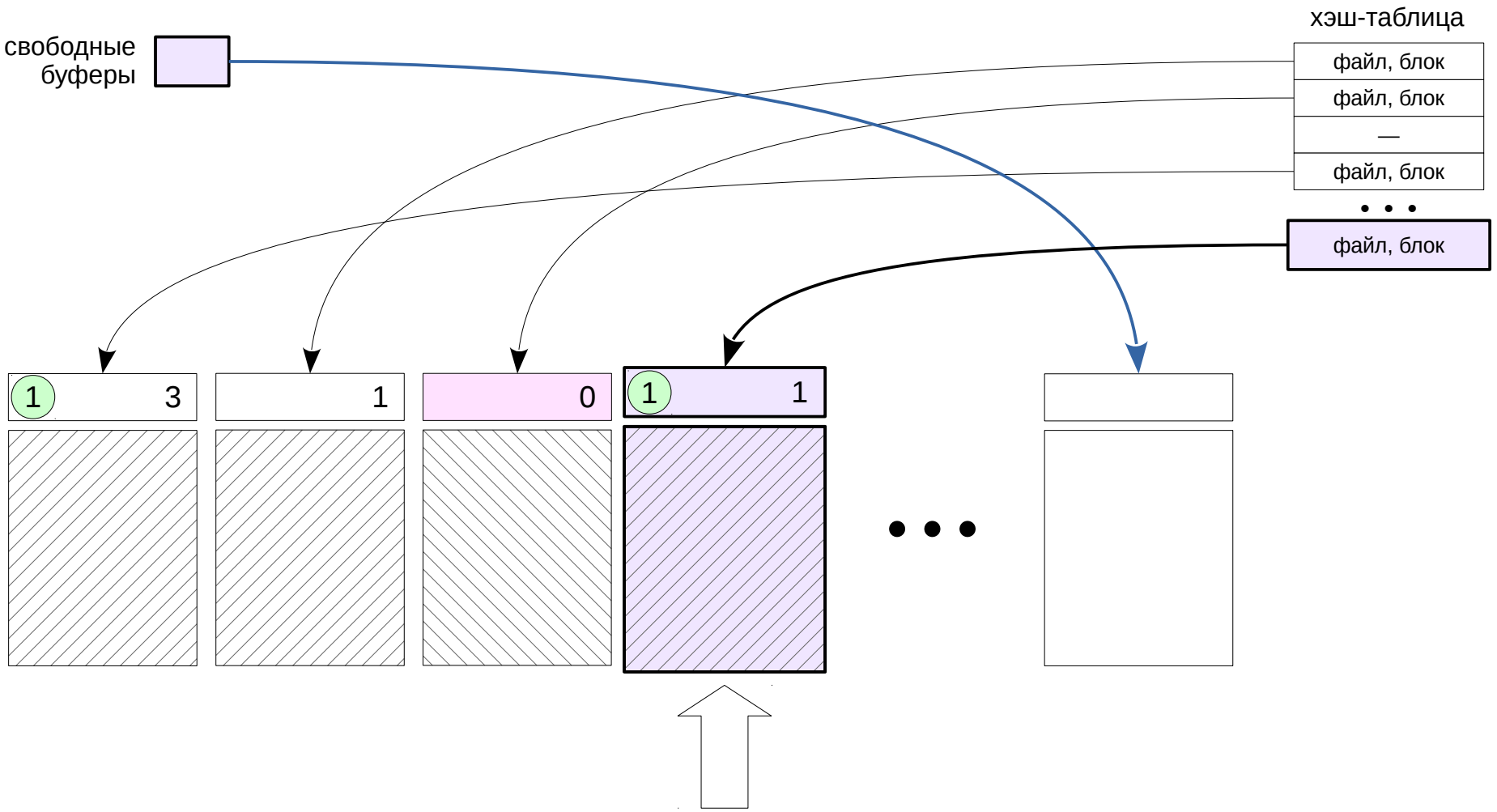
Чтение страницы из кэша



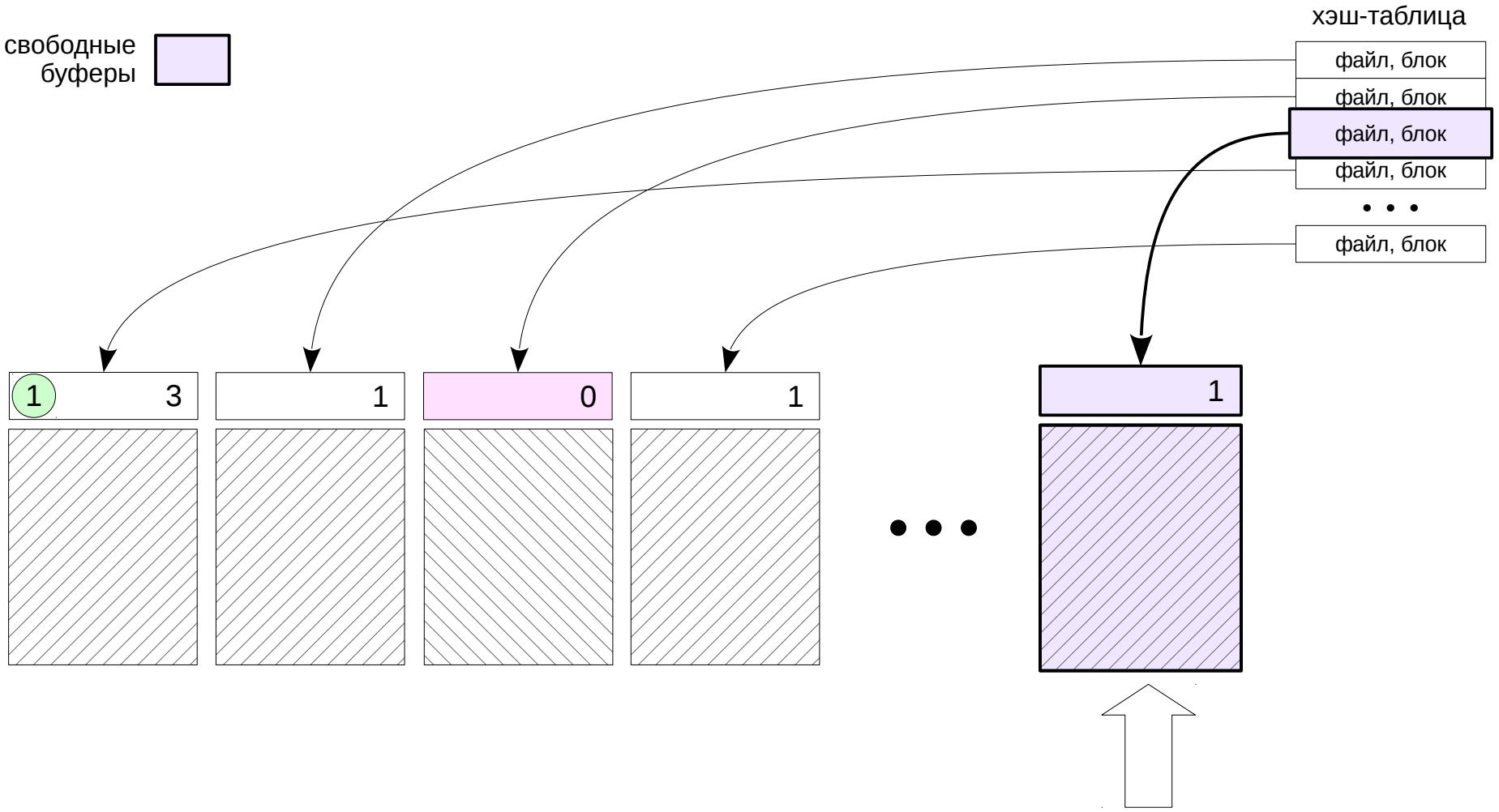
Чтение страницы с диска



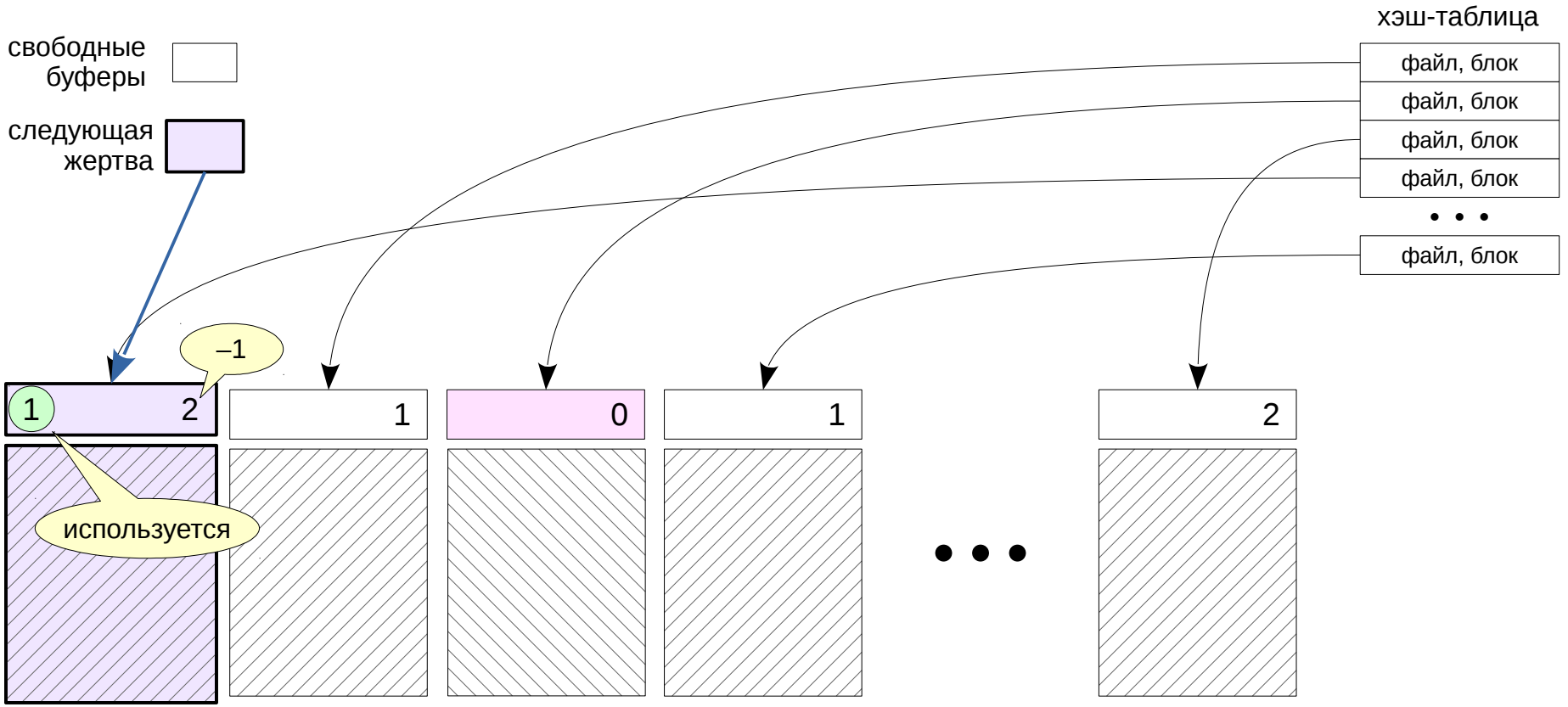
Чтение страницы с диска



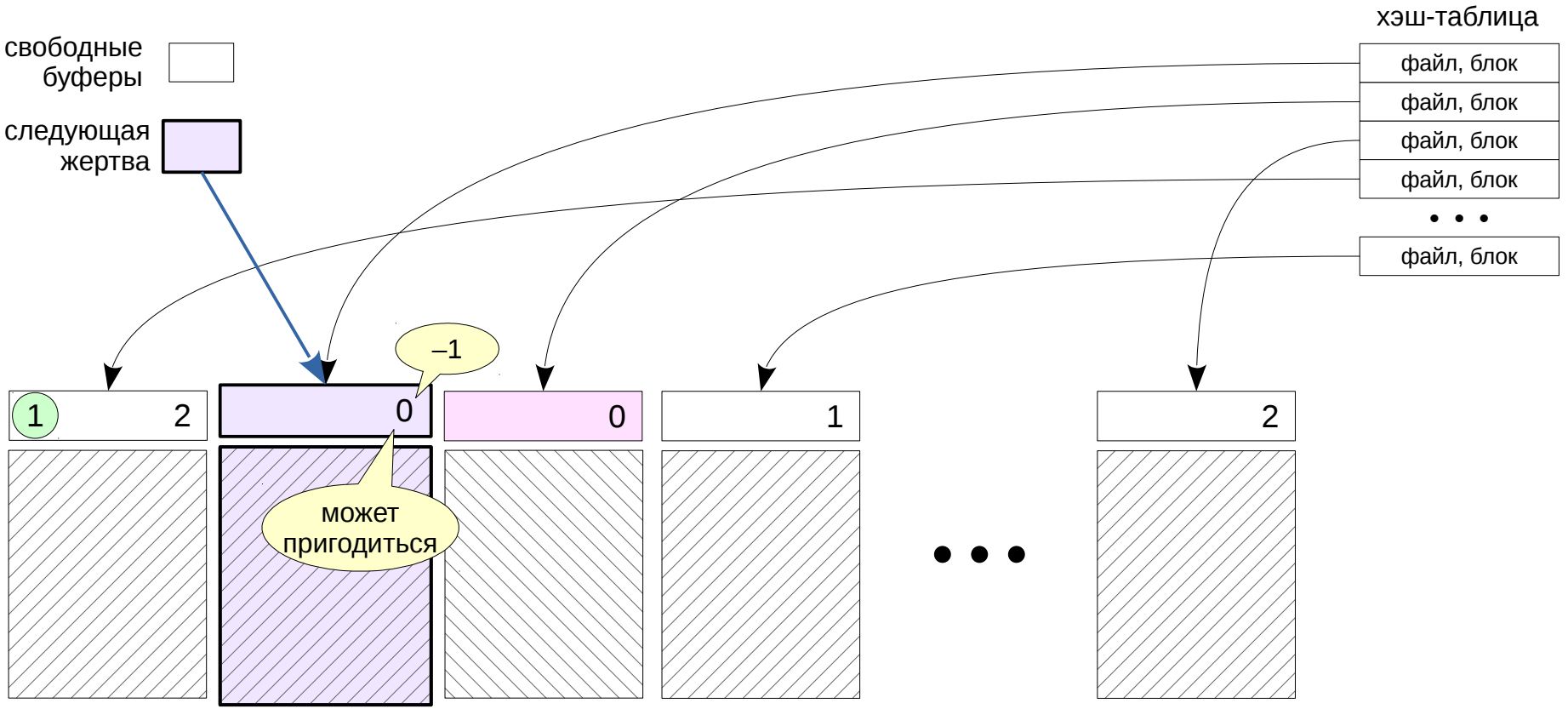
Чтение страницы с диска



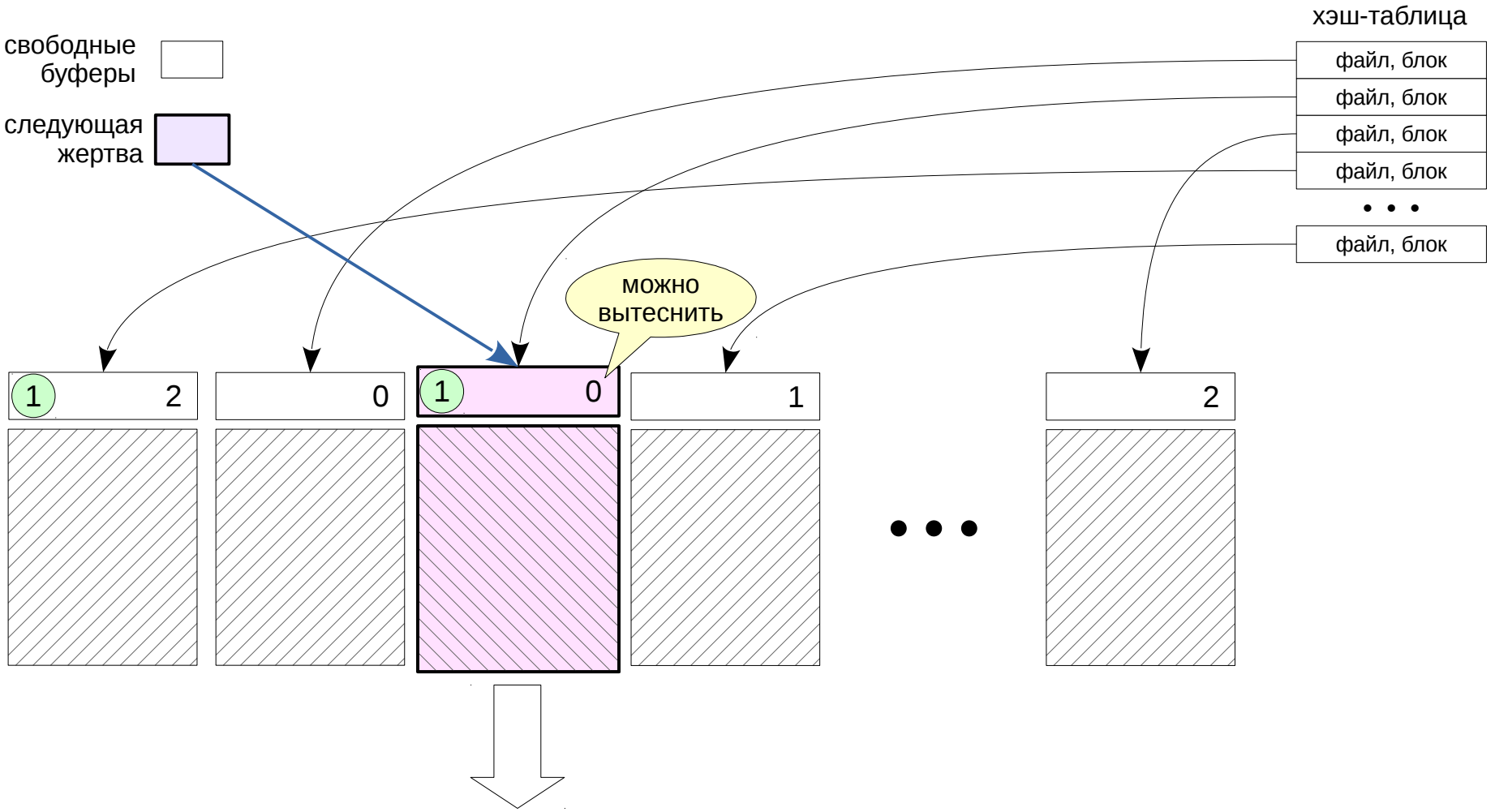
Чтение с вытеснением



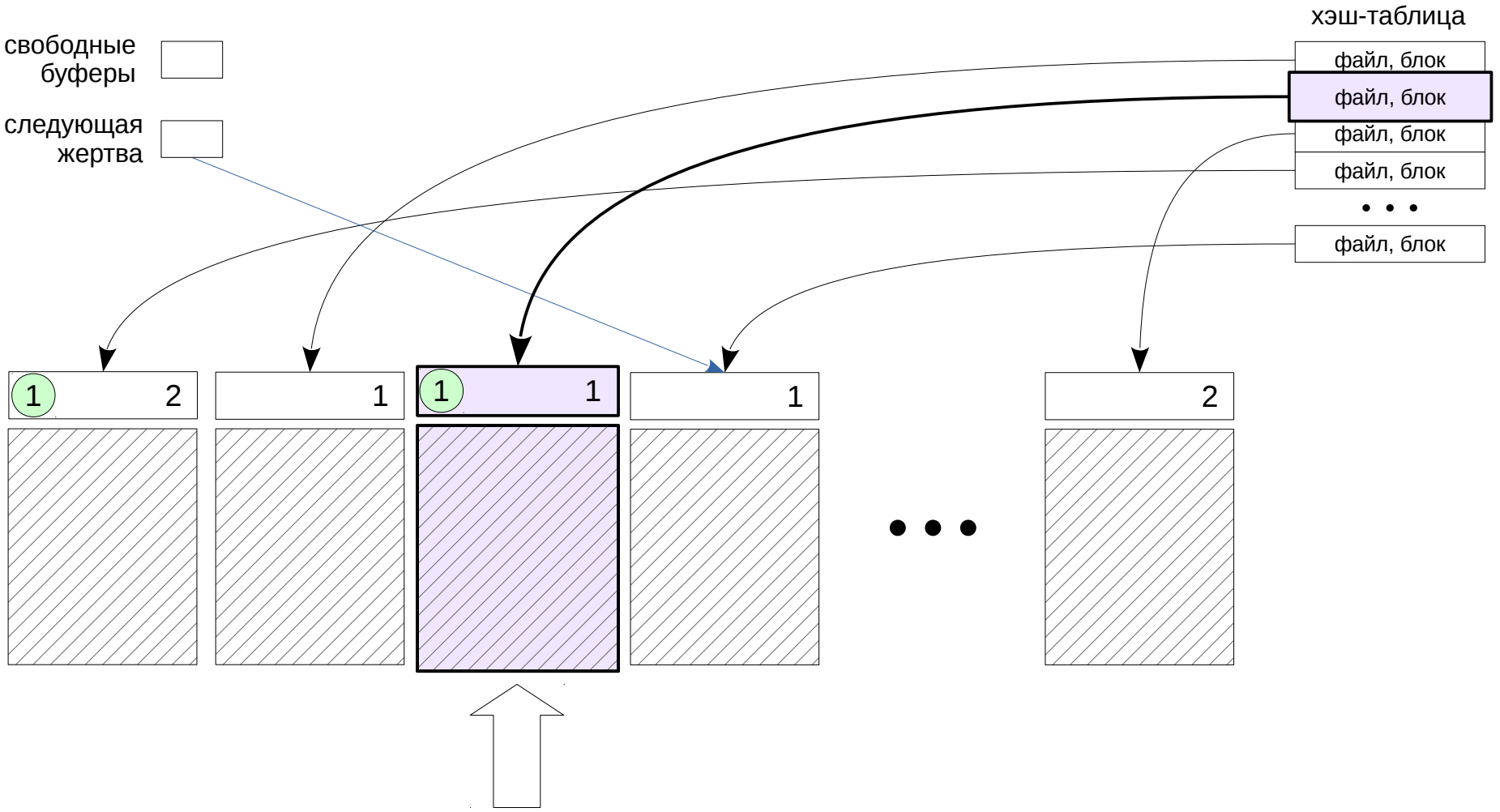
Чтение с вытеснением



Чтение с вытеснением



Чтение с вытеснением



Блокировки в памяти

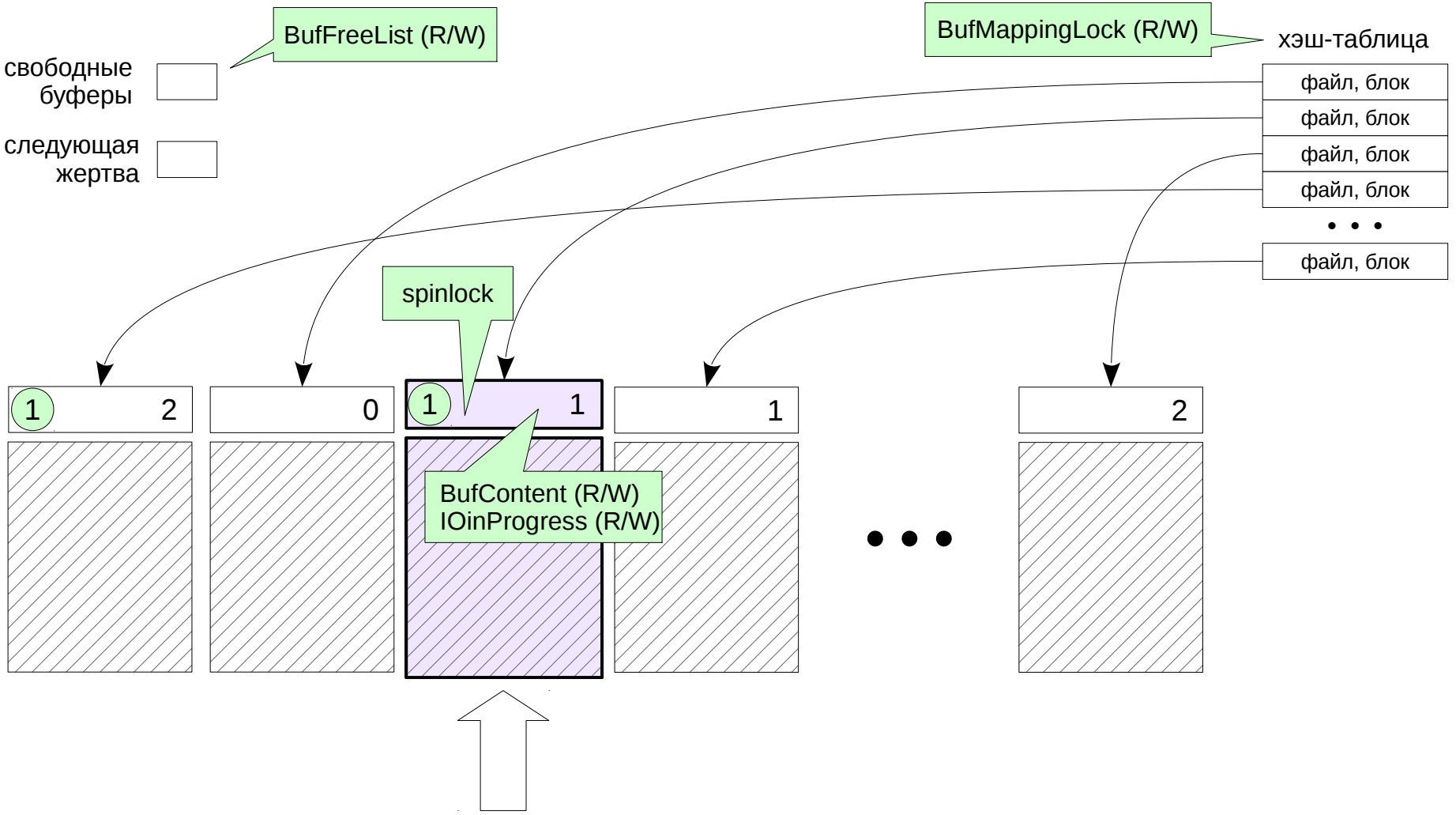
Спин-блокировки

- очень короткое время
- цикл активного ожидания

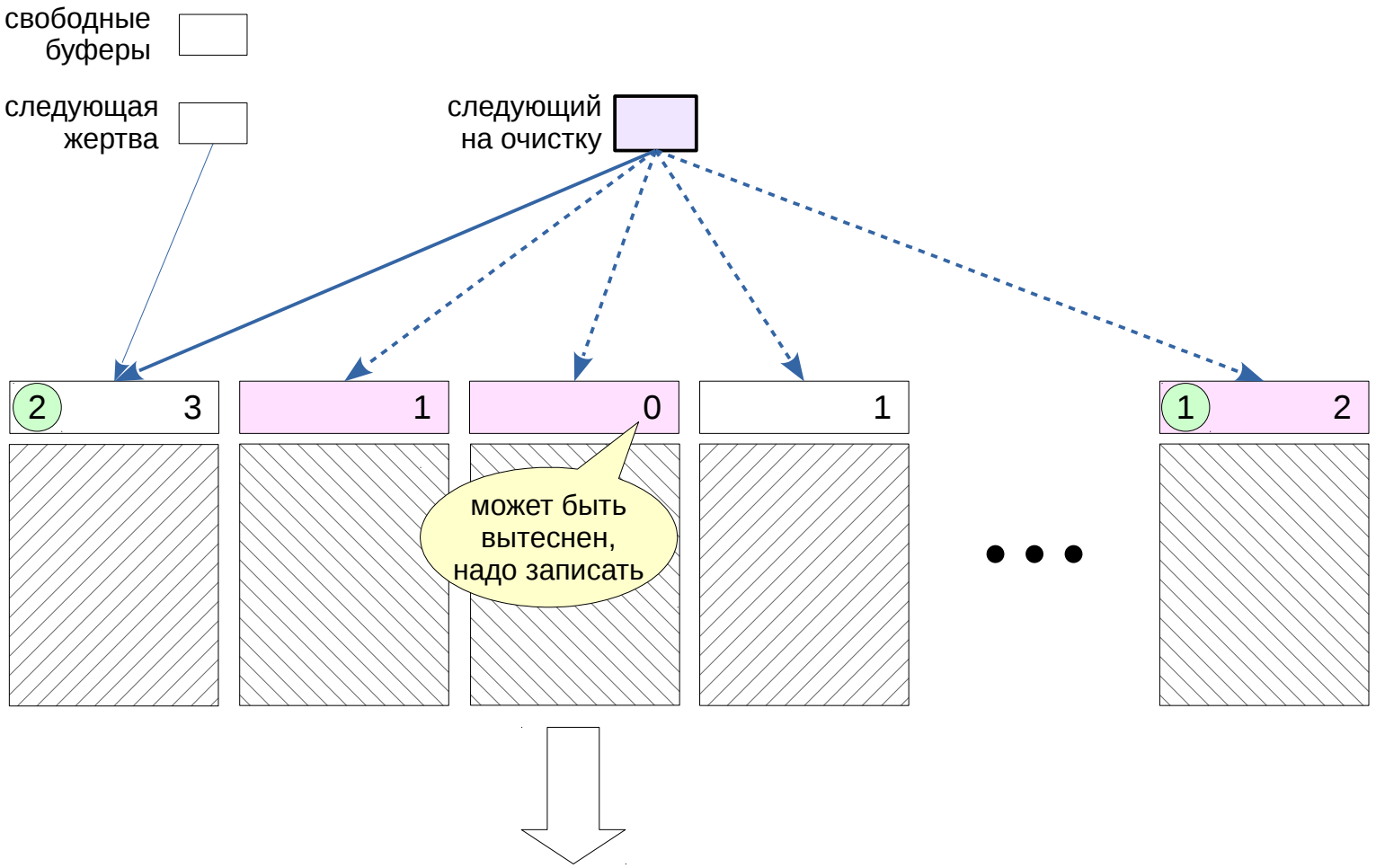
Легковесные блокировки

- короткое время
- эксклюзивные (запись) или разделяемые (чтение)
- очередь ждущих
- нет обнаружения взаимоблокировок

Блокировки в памяти



Процесс фоновой записи



Процесс фоновой записи

Алгоритм

записать столько грязных буферов, сколько буферов было запрошено серверными процессами с прошлого раза * *bgwriter_lru_multiplier*
(но не больше *bgwriter_lru_maxpages*)

уснуть на *bgwriter_delay*
(но если совсем не было работы, то дольше)

Настройки

bgwriter_delay	= 200ms
bgwriter_lru_maxpages	= 100
bgwriter_lru_multiplier	= 2.0

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

Временные таблицы

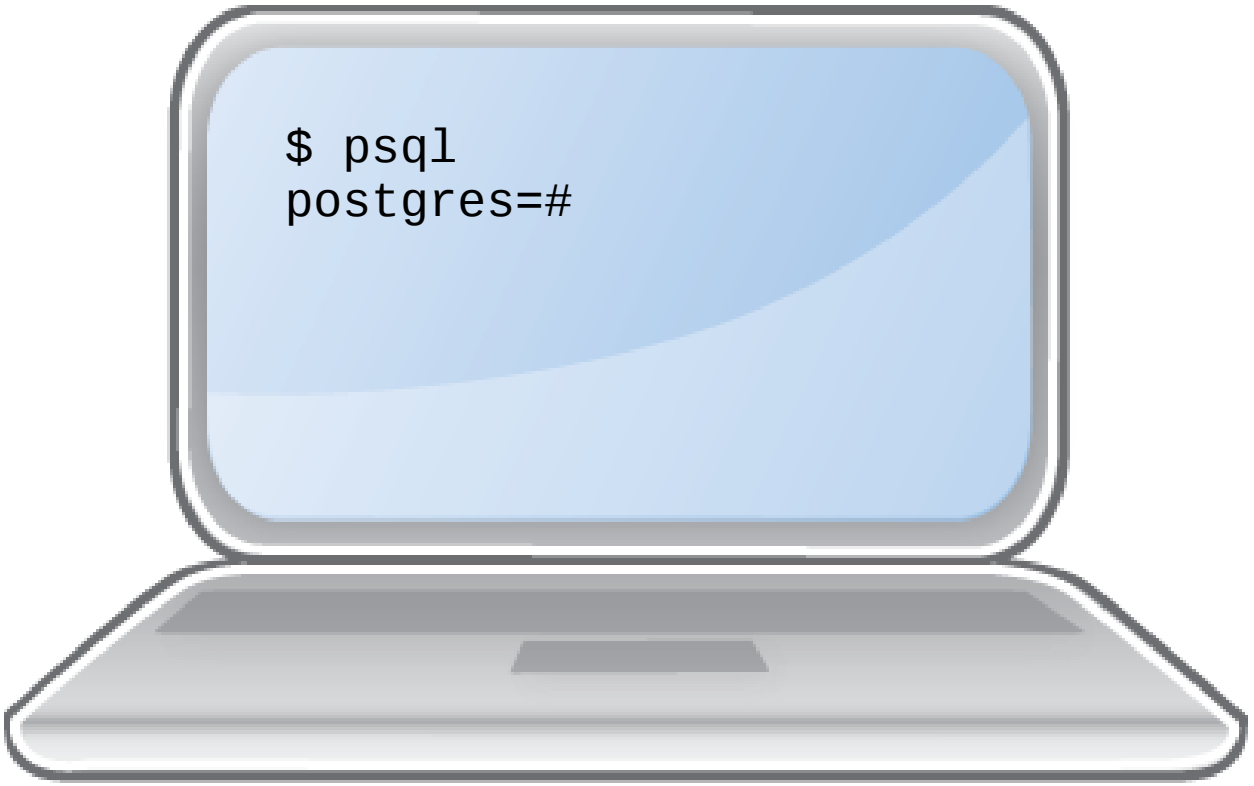
Данные временных таблиц

- видны только одному сеансу
- не попадают в WAL
- не требуют контрольной точки

Используется локальный буферный кэш

- не требуются блокировки
- память выделяется по необходимости в пределах *temp_buffers*
- обычный алгоритм вытеснения, но без фонового режима

Демонстрация



Настройка размера кэша

Буферный кэш должен содержать «активные» данные

- при меньшем размере постоянно вытесняются полезные страницы (признак: все страницы имеют большое число обращений)
- при большем размере бессмысленно растут накладные расходы

Кэширование на уровне операционной системы

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

Прогрев буферного кэша и кэша ОС

- требует времени после перезапуска сервера
- для ускорения — pg_prewarm

Итоги

Вся работа с данными происходит через буферный кэш

Редко используемые страницы вытесняются,
часто используемые — остаются

Одновременный доступ обеспечивается блокировками

Практика

1. Создайте БД DB7 и в ней таблицу T с 10 000 строками.
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.
5. Сколько грязных буферов осталось теперь?

Авторы: Егор Рогов, Павел Лузанов

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

Темы

Блокировки в памяти

Устройство и использование буферного кэша

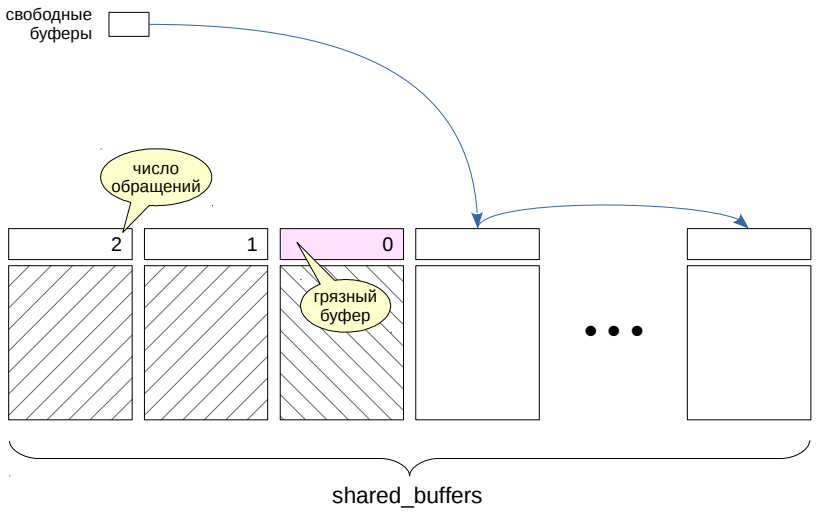
Механизм вытеснения

Запись грязных буферов

Процесс фоновой записи (writer)

Настройка размера кэша

Чтение страницы из кэша



Буферный кэш представляет собой набор буферов — мест в памяти, в которые можно прочитать с диска страницу и работать с ней, — доступных всем процессам. Кроме собственно страницы буфер содержит дополнительную информацию:

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

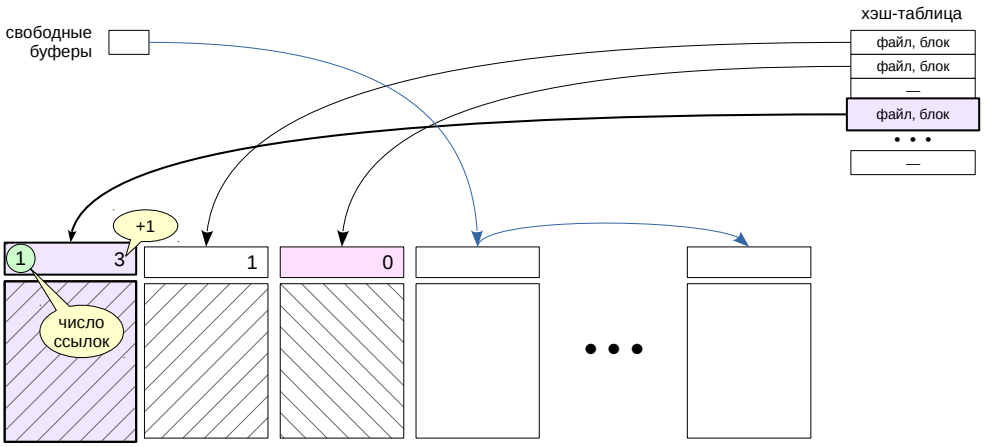
- число обращений к буферу (увеличивается каждый раз, когда к буферу происходит обращение; максимальное значение — 5)

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

Размер буферного кэша задается параметром shared_buffers.

Изначально кэш содержит пустые буферы, и все они связаны в список свободных буферов.

Чтение страницы из кэша

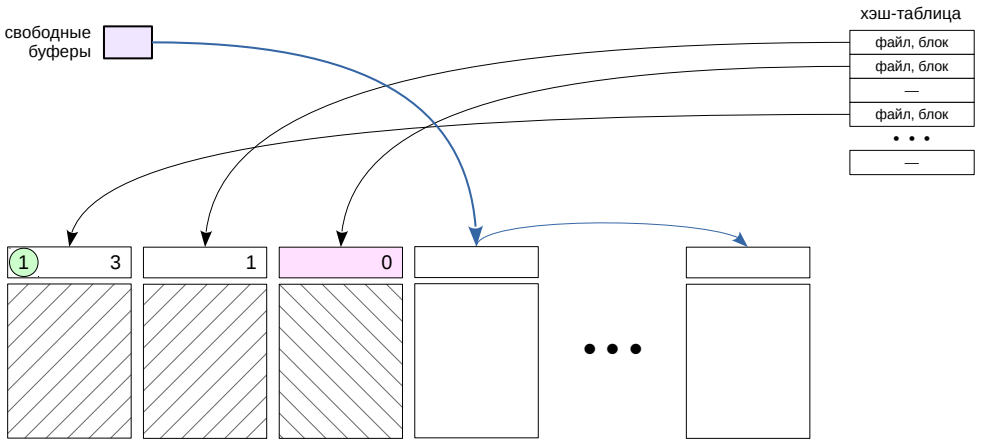


Когда процессу требуется прочитать страницу, он сначала пытается найти ее в буферном кэше. Для этого существует хэш-таблица, ключами в которой служат адреса страниц (файл и номер страницы внутри файла), а значениями — номера буферов.

Если процесс нашел нужную страницу в кэше, он должен увеличить число ссылок на нее (pin count) и число обращений (usage count).

Пока число ссылок больше нуля, считается, что буфер используется и его содержимое не должно «радикально» измениться. Например, в странице может появиться новая версия строки (поскольку ее доступность для других транзакций регламентирована правилами видимости), но в буфер не может быть прочитана другая страница.

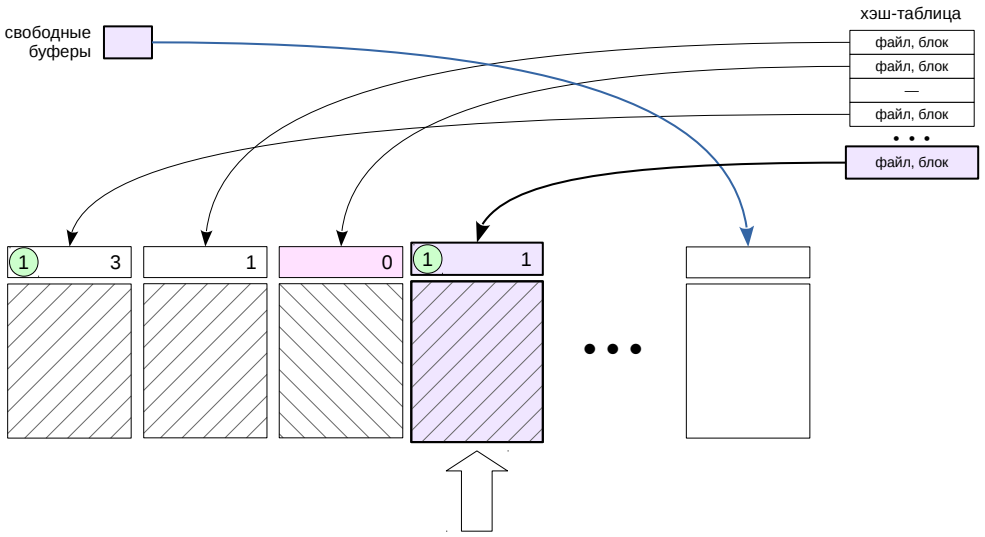
Чтение страницы с диска



Может получиться так, что необходимая страница не будет найдена в кэше. В этом случае ее необходимо считать с диска, а для этого требуется новый буфер.

В первую очередь поиск нового буфера будет проходить с помощью списка свободных буферов.

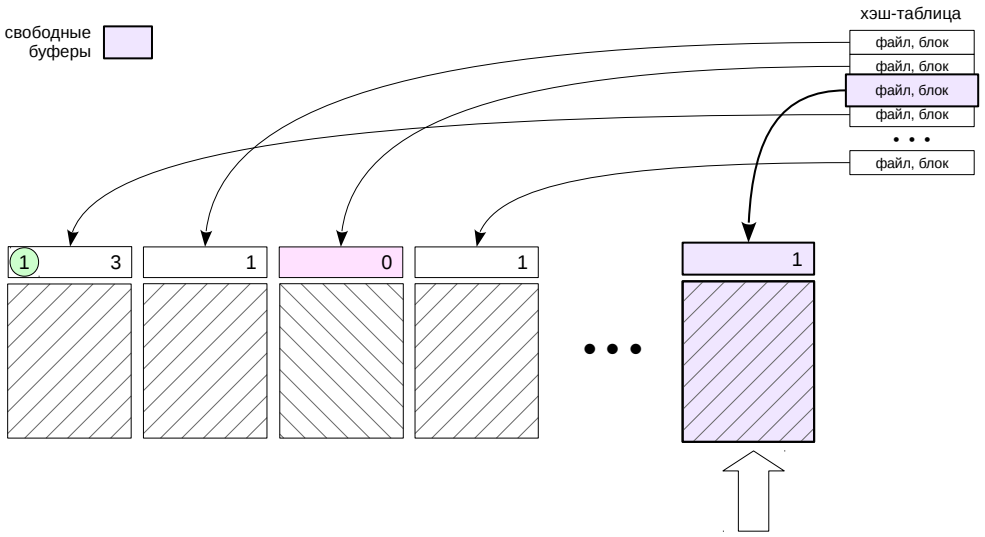
Чтение страницы с диска



В найденном по указателю буфере увеличивается число ссылок и в него читается необходимая страница.

Ссылку на загруженную страницу необходимо прописать в хэш-таблицу, чтобы в дальнейшем ее можно было найти.

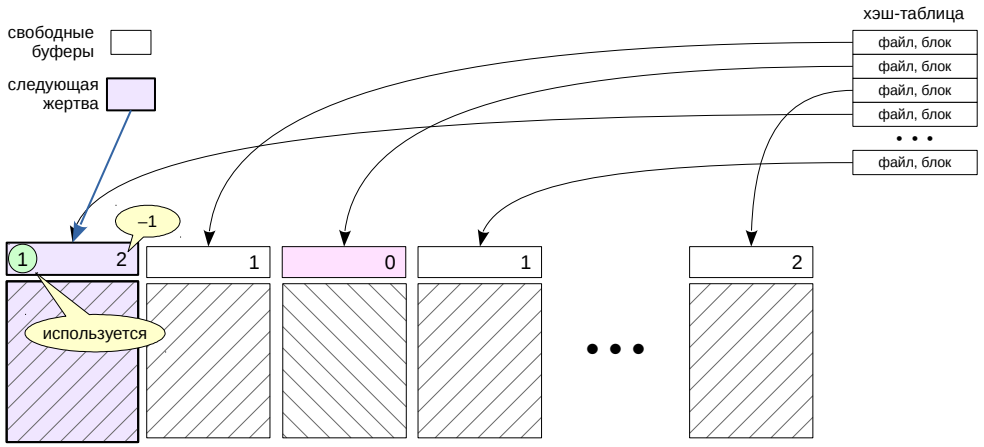
Чтение страницы с диска



Таким же образом будет заполнен и последний свободный буфер.

Теперь ссылка на свободный буфер пуста — в дальнейшем, чтобы прочитать новую страницу, какую-то другую придется вытеснять из буфера.

Чтение с вытеснением



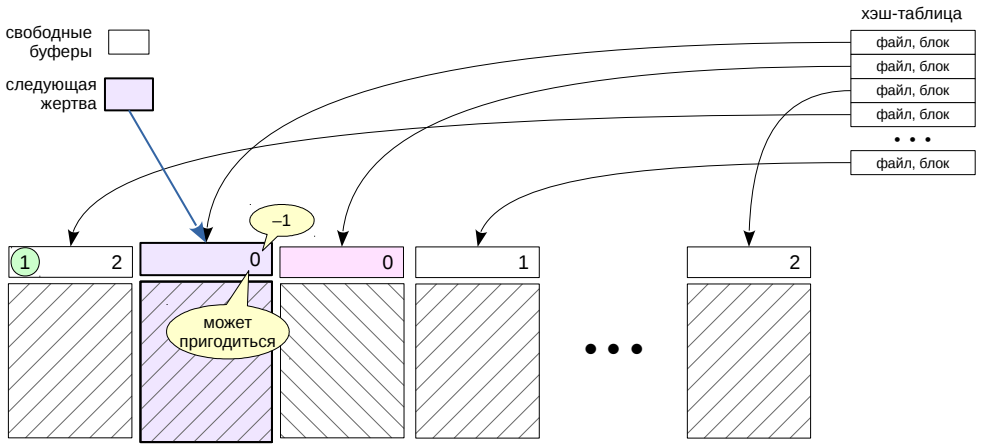
Механизм вытеснения использует еще один указатель — на следующую «жертву».

Алгоритм состоит в том, чтобы перебирать по кругу все буферы, уменьшая их счетчик обращений (usage count). Вытесняется страница из того буфера, у которого первым окажется нулевое значение. При этом у буферов, которые используются чаще других, больше шансов задержаться в кэше. По-английски этот алгоритм называется clock-sweep.

Максимальное значение счетчика обращений ограничено числом 5, чтобы избежать «наматывания кругов» при больших значениях.

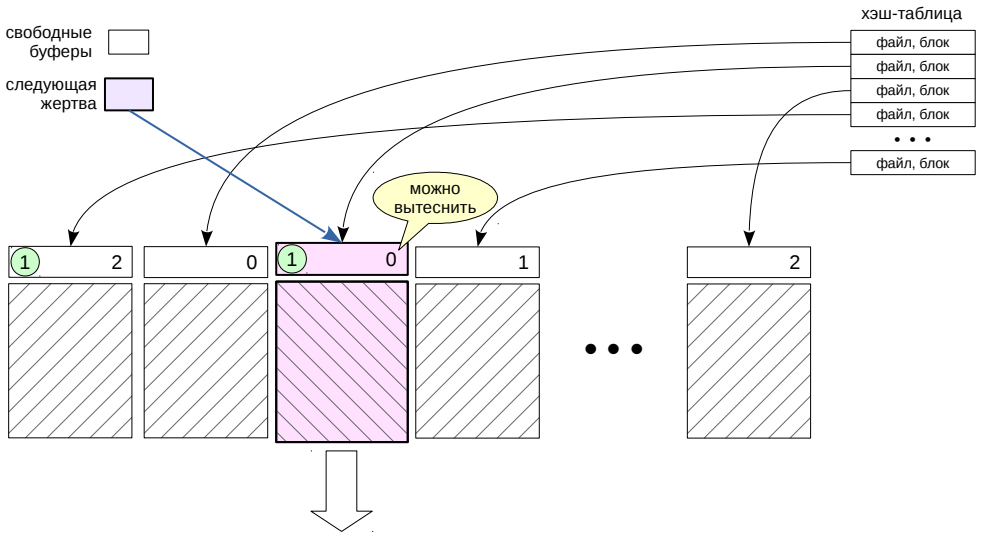
В нашем примере процесс обращается к буферу по указателю. Буфер имеет ненулевой счетчик ссылок, то есть используется каким-либо процессом. Поэтому он не может быть вытеснен. Значение счетчика обращений уменьшается на единицу и мы переходим к следующему буферу.

Чтение с вытеснением



Следующий буфер не используется, но его счетчик обращений ненулевой. Он уменьшается на единицу.

Чтение с вытеснением

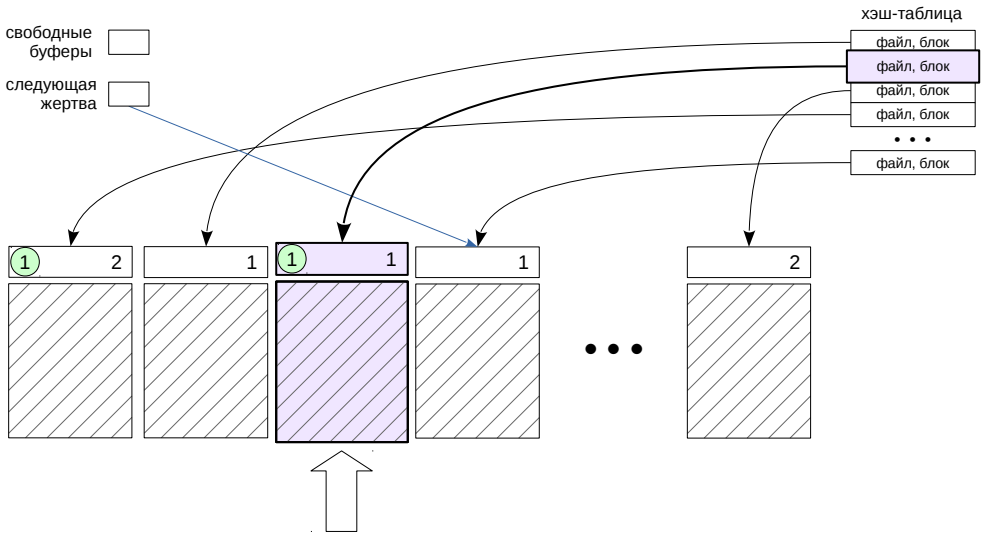


Наконец, мы приходим к третьему буферу с нулевым счетчиком обращений и нулевым счетчиком ссылок. Страница из этого буфера и будет вытеснена.

Однако в данном случае буфер оказался грязным, поэтому страницу требуется сохранить, прежде чем ее можно будет заменить другой. Для этого увеличивается счетчик ссылок (чтобы показать, что буфер используется), после чего страница записывается на диск.

Именно с целью уменьшить задержку при чтении страницы и сделан фоновый процесс записи, который старается брать на себя задачу своевременной записи грязных буферов. Этот процесс рассмотрен ниже.

Чтение с вытеснением



Затем уже знакомым образом в освободившийся буфер читается новая страница.

Заметим, что ссылка на «жертву» указывает на следующий буфер. У только что загруженного буфера есть время нарастить счетчик обращений, пока указатель не пройдет по всему буферному кэшу и не вернется вновь.

Блокировки в памяти

Спин-блокировки

- очень короткое время
- цикл активного ожидания

Легковесные блокировки

- короткое время
- эксклюзивные (запись) или разделяемые (чтение)
- очередь ждущих
- нет обнаружения взаимоблокировок

Ранее рассматривались так называемые тяжеловесные блокировки. Они предназначены для захвата на продолжительное время (например, в течение всей транзакции или сеанса), поддерживают множество режимов и обеспечивают очередь ожидающих с обнаружением взаимоблокировок.

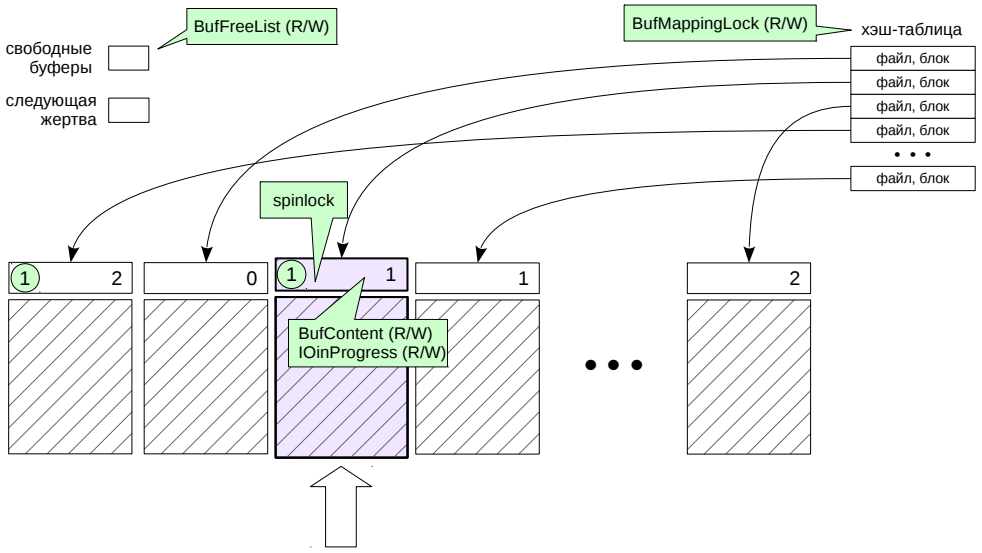
Для защиты структур в оперативной памяти, разделяемой несколькими процессами, используются более простые (и дешевые в смысле накладных расходов) блокировки.

Самые простые блокировки — spinlock. Они предназначена для захвата на очень короткое время (несколько инструкций) и защищают отдельные поля от одновременного изменения.

Следом идут так называемые легковесные блокировки. Их захватывают на короткое время, которое требуется для работы со структурой данных (например, с хэш-таблицей или списком указателей). Поддерживаются два режима: эксклюзивный (для записи) и разделяемый (для чтения). Соответственно обеспечивается очередь ожидающих, но проверка взаимоблокировок не выполняется.

Далее мы рассмотрим использование этих блокировок на примере буферного кэша.

Блокировки в памяти



Чтобы обратиться к хэш-таблице, содержащей ссылки на буферы, процесс должен захватить легковесную блокировку BufMappingLock в разделяемом режиме (иначе в процессе чтения данные могут быть изменены другим процессом), а если таблицу требуется изменять — то в эксклюзивном режиме. На самом деле существует несколько блокировок, каждая из которых защищает свою часть таблицы — это сделано для того, чтобы уменьшить конкуренцию.

Доступ к заголовку буфера (например, чтобы изменить счетчик ссылок) процесс получает с помощью спин-блокировки.

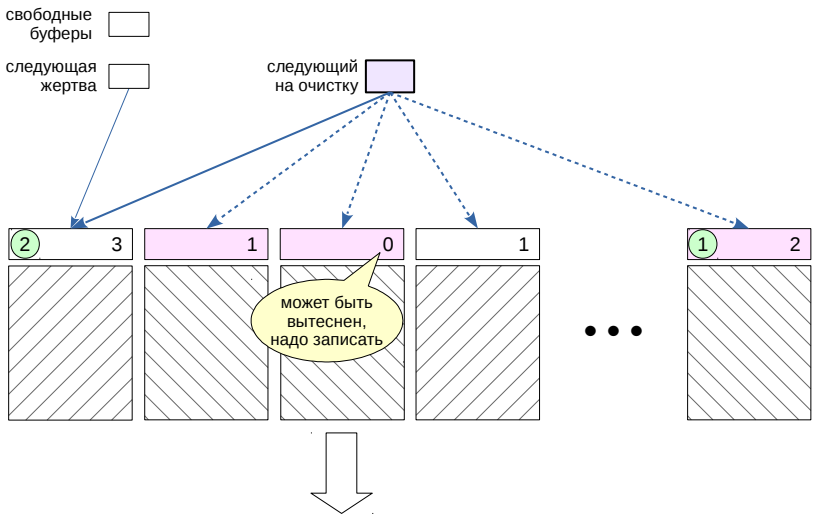
Чтобы прочитать содержимое буфера, требуется блокировка BufContent. Обычно она захватывается только для чтения указателей на версии строк, а дальше достаточно защиты, предоставляемой счетчиком ссылок. Для изменения содержимого буфера эта блокировка захватывается в эксклюзивном режиме.

При чтении буфера с диска (или записи на диск) захватывается таже блокировка IO_in_Progress, которая сигнализирует другим процессам, что страница читается — они могут встать в очередь, если им тоже нужна та же самая страница.

Указатели на свободные буферы и на следующую «жертву» защищены одной легковесной блокировкой BufFreeList.

Как можно видеть, работа с буферным кэшем не обходится даром, хотя разработчики PostgrSQL прилагают постоянные усилия к тому, чтобы снизить накладные расходы и уменьшить конкуренцию за ресурсы.

Процесс фоновой записи



Процесс фоновой записи использует тот же самый алгоритм поиска буферов для вытеснения, только использует свой указатель. Он может опережать указатель на «жертву», но не отставать от него.

Записываются буферы, которые:

- содержат измененные данные (грязные),

- с нулевым числом ссылок (pin count),

- и с нулевым числом обращений (usage count).

Таким образом фоновый процесс записи находит те буферы, которые с большой вероятностью вскоре потребуется вытеснить. За счет этого серверный процесс, которому требуется новый буфер для чтения или записи данных, с большой вероятностью обнаружит его не грязным.

Процесс фоновой записи

Алгоритм

записать столько грязных буферов, сколько буферов было запрошено серверными процессами с прошлого раза * *bgwriter_lru_multiplier*
(но не больше *bgwriter_lru_maxpages*)

уснуть на *bgwriter_delay*
(но если совсем не было работы, то дольше)

Настройки

bgwriter_delay	= 200ms
bgwriter_lru_maxpages	= 100
bgwriter_lru_multiplier	= 2.0

Процесс фоновой записи работает порциями максимум по bgwriter_lru_maxpages, засыпая после каждой на bgwriter_delay.

Таким образом, если установить параметр bgwriter_lru_maxpages в ноль, процесс фактически не будет работать.

Точное значение буферов для записи определяется по среднему количеству буферов, которые запрашивались серверными процессами с прошлого запуска (используется скользящее среднее, чтобы сгладить неравномерность между запусками, но при этом не зависеть от старой истории). Вычисленное количество буферов умножается на коэффициент bgwriter_lru_multiplier.

Если процесс совсем не обнаружил грязных буферов (то есть в системе ничего не происходит), он «впадает в спячку», из которой его выводит обращение серверного процесса за буфером. После этого процесс просыпается и опять работает обычным образом.

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

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, используется так называемое буферное кольцо — часть буферного кэша, выделенная для одной операции. При этом вытеснение действует в пределах кольца, поэтому остальные данные в буфере не страдают.

Чем больше кольцо, тем больше времени пройдет до того, как потребуется вытеснение. Это особенно важно в случае, когда блоки меняются и, следовательно, должны записываться на диск. При большом размере кольца есть шанс, что буферы будут успевать записываться процессом background writer. С другой стороны, меньше страниц остается в основном кэше.

Для последовательного чтения (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.

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 кольцо имеет достаточно большой размер (но не больше одной восьмой всего буферного кэша).

Временные таблицы

Данные временных таблиц

- видны только одному сеансу
- не попадают в WAL
- не требуют контрольной точки

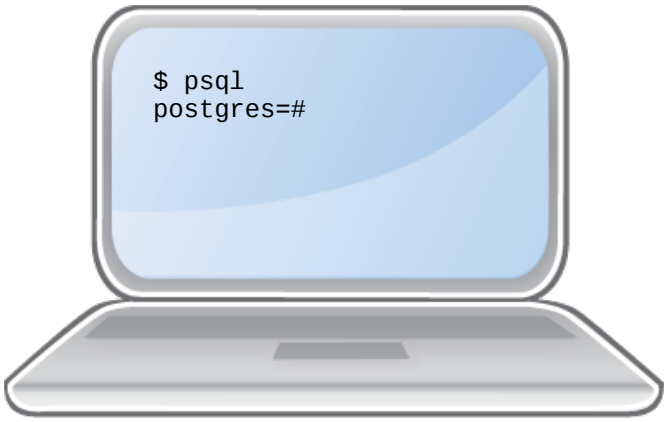
Используется локальный буферный кэш

- не требуются блокировки
- память выделяется по необходимости в пределах *temp_buffers*
- обычный алгоритм вытеснения, но без фонового режима

Для временных таблиц используется оптимизация. Поскольку временные данные видны только одному процессу, им нечего делать в общем буферном кэше. Для них используется облегченный локальный кэш.

Поскольку кэш локальный, 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, но фоновый процесс writer не участвует в записи страниц.

Демонстрация



Настройка размера кэша

Буферный кэш должен содержать «активные» данные

- при меньшем размере постоянно вытесняются полезные страницы (признак: все страницы имеют большое число обращений)
- при большем размере бессмысленно растут накладные расходы

Кэширование на уровне операционной системы

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

Прогрев буферного кэша и кэша ОС

- требует времени после перезапуска сервера
- для ускорения — pg_prewarm

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 кэш еимеют большое число обращений (usage count).

При большем размере кэша накладные расходы на его поддержку будут уменьшать получаемые преимущества.

Не следует забывать и о том, что Постгрес 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, отсутствующая в буферном кэше страница не всегда приводит к необходимости реального ввода-вывода. При достаточном размере кэше ОС страница может обнаружиться там. Стратегия вытеснения ОС не учитывает специфики баз данных: например, полное чтение таблицы не вытесняет буферный кэш, но вытесняет кэш ОС.

При перезапуске сервера должно пройти определенное время, пока оба кэша заполнятся полезными данными. Это время можно уменьшить, используя предварительный «прогрев», например, с помощью расширения pg_prewarm.

Итоги

Вся работа с данными происходит через буферный кэш

Редко используемые страницы вытесняются,
часто используемые — остаются

Одновременный доступ обеспечивается блокировками

Практика

1. Создайте БД DB7 и в ней таблицу T с 10 000 строками.
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.
5. Сколько грязных буферов осталось теперь?

Чтобы проверить содержимое буферного кэша, используйте расширение pg_buffercahce.

Нагрузку удобно имитировать с помощью расширения pgbench.