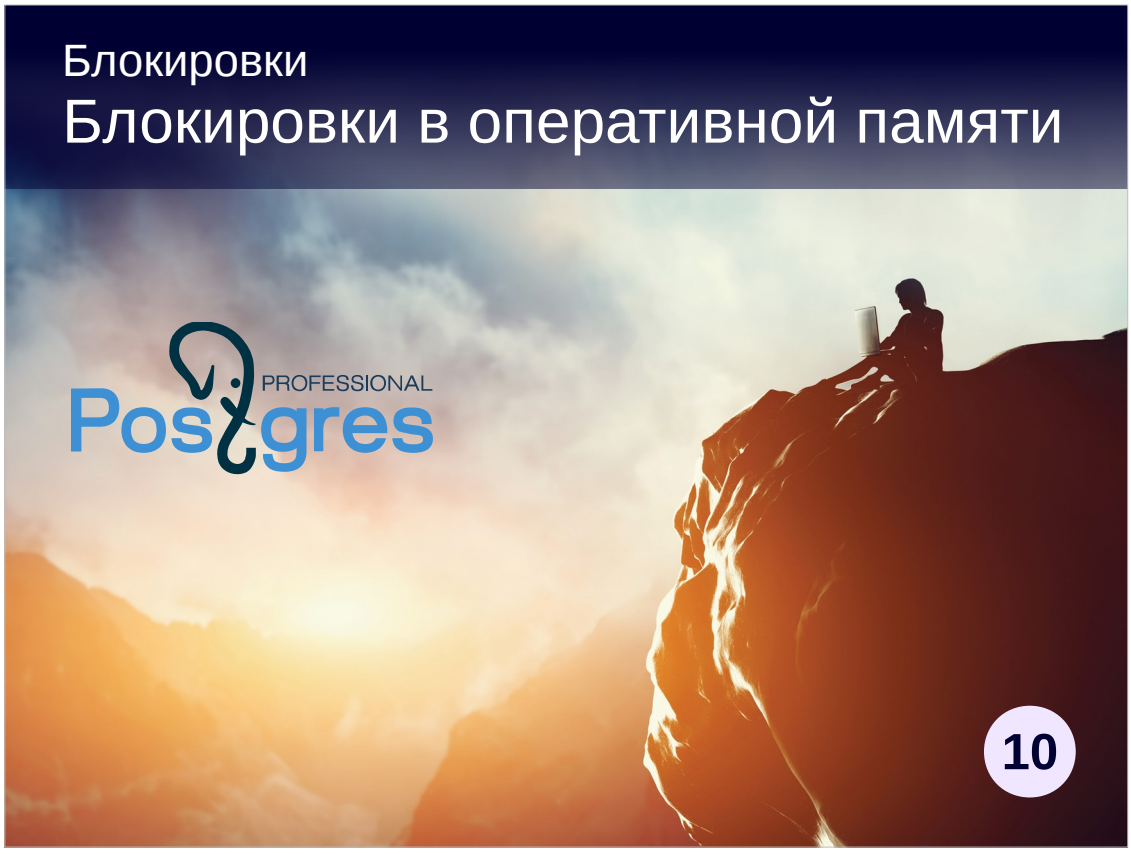


Авторы: Егор Рогов, Павел Лузанов

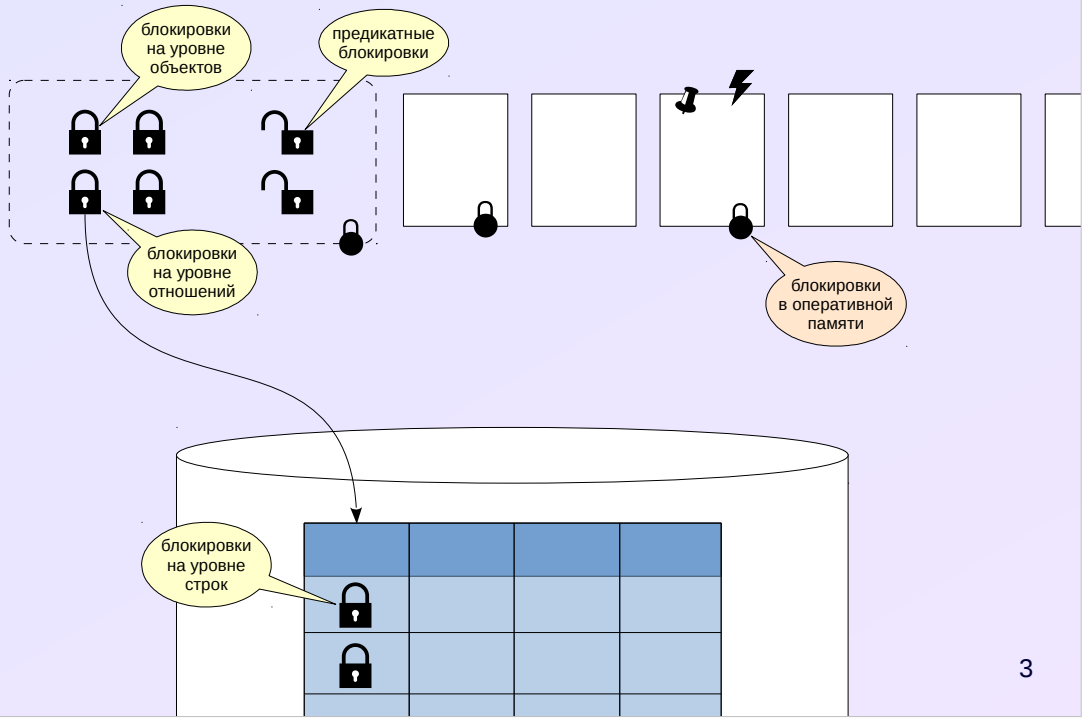
Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Блокировки в памяти

Мониторинг ожиданий

Блокировки в памяти



Спин-блокировки

- Устанавливаются на очень короткое время
  - несколько инструкций процессора
- Единственный режим — исключительный
- Нет возможности мониторинга
- Нет обнаружения взаимоблокировок
- Цикл активного ожидания
  - используются атомарные инструкции процессора

Мы уже рассмотрели «тяжелые» блокировки, действующие как правило до конца транзакции и поддерживающие множество режимов. Для защиты структур в оперативной памяти, разделяемой несколькими процессами, используются более простые (и дешевые в смысле накладных расходов) блокировки.

Самые простые из них — спин-блокировки или спинлоки (spinlock). Они предназначены для захвата на очень короткое время (несколько инструкций процессора) и защищают отдельные поля от одновременного изменения.

Спин-блокировки реализуются на основе атомарных инструкций процессора, например, compare-and-swap. Они захватываются только в исключительном режиме. Если блокировка занята, выполняется цикл активного ожидания — команда повторяется до тех пор, пока не выполнится успешно. Это имеет смысл, поскольку спин-блокировки применяются только в тех случаях, когда вероятность конфликта очень мала.

Спин-блокировки не обеспечивают обнаружения взаимоблокировок (за этим следят разработчики PostgreSQL) и не предоставляют никаких средств мониторинга. По большому счету, единственное, что мы можем сделать со спин-блокировками — знать о их существовании.

Легкие блокировки

- Устанавливаются на короткое время
  - обычно доли секунды
- Исключительный и разделяемый режимы
- Есть мониторинг
- Нет обнаружения взаимоблокировок
- Пассивное ожидание
  - при освобождении ресурса возникает состояние гонки, выигрывает случайный процесс

Следом идут так называемые легкие блокировки (lightweight locks, lwlocks).

Их захватывают на короткое время, которое требуется для работы со структурой данных (например, с хеш-таблицей или списком указателей). Как правило, легкая блокировка удерживается недолго, но в процессе ее удержания могут выполняться операции ввода-вывода, так что время может оказаться и значительным.

Поддерживаются два режима: исключительный (для записи) и разделяемый (для чтения). Как таковой очереди ожидания нет: если несколько процессов ждут освобождения блокировки, ее получит один из них более или менее случайным образом. В системах с высокой параллельностью и большой нагрузкой это может приводить к неприятным эффектам (https://postgrespro.ru/list/thread-id/2400193).

Проверка взаимоблокировок не выполняется, как и для спин-блокировок.

Однако легкие блокировки имеют средства для мониторинга.

Закрепление буфера

- Устанавливается на время работы с буфером
  - возможно длительное
- Исключительный и разделяемый режимы
- Есть мониторинг
- Есть обнаружение взаимоблокировок
- Пассивное ожидание
  - но обычно закрепленный буфер пропускается

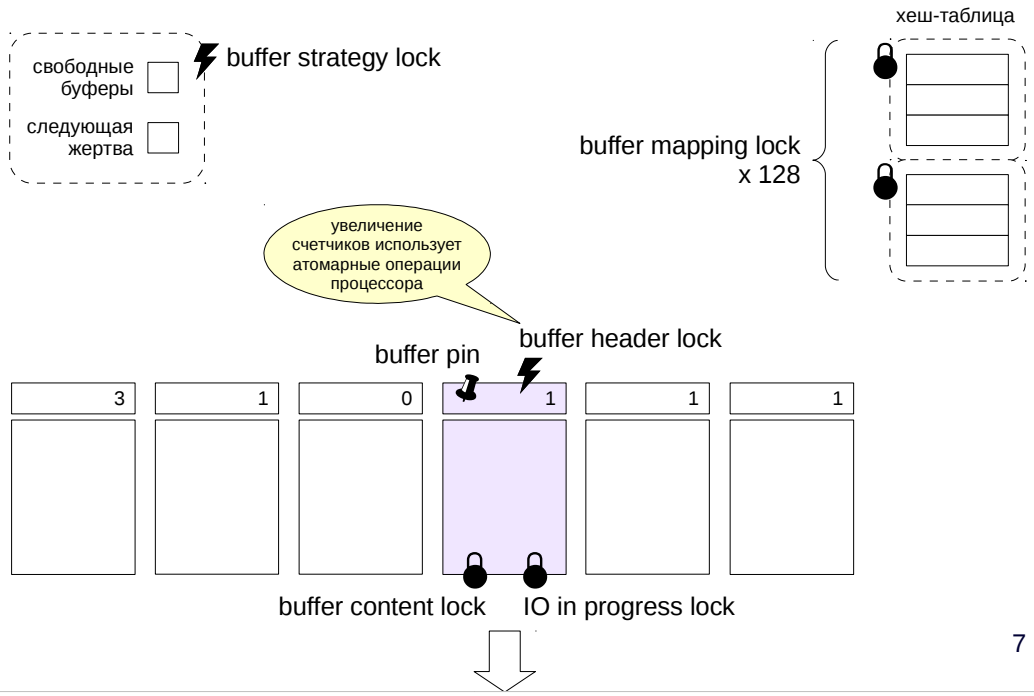
Еще один вид блокировки, который мы уже рассматривали в теме «Буферный кэш» модуля «Журналирование» — закрепление буфера (buffer pin).

С закрепленным буфером можно выполнять только некоторые действия. Например, в страницу можно вставить новую строку (которую остальные процессы не увидят благодаря многоверсионности), но нельзя заменить одну страницу на другую.

Как правило, процессы пропускают закрепленный буфер и выбирают другой, чтобы не ждать снятия закрепления. Но в некоторых случаях, когда требуется именно этот буфер, выполняется пассивное ожидание — система «будит» ожидающий процесс, когда закрепление снимается.

Ожидания, связанные с закреплением, доступны для мониторинга.

Пример: буферный кэш



Чтобы получить некоторое (неполное) представление о том, как и где используются блокировки, рассмотрим пример буферного кэша.

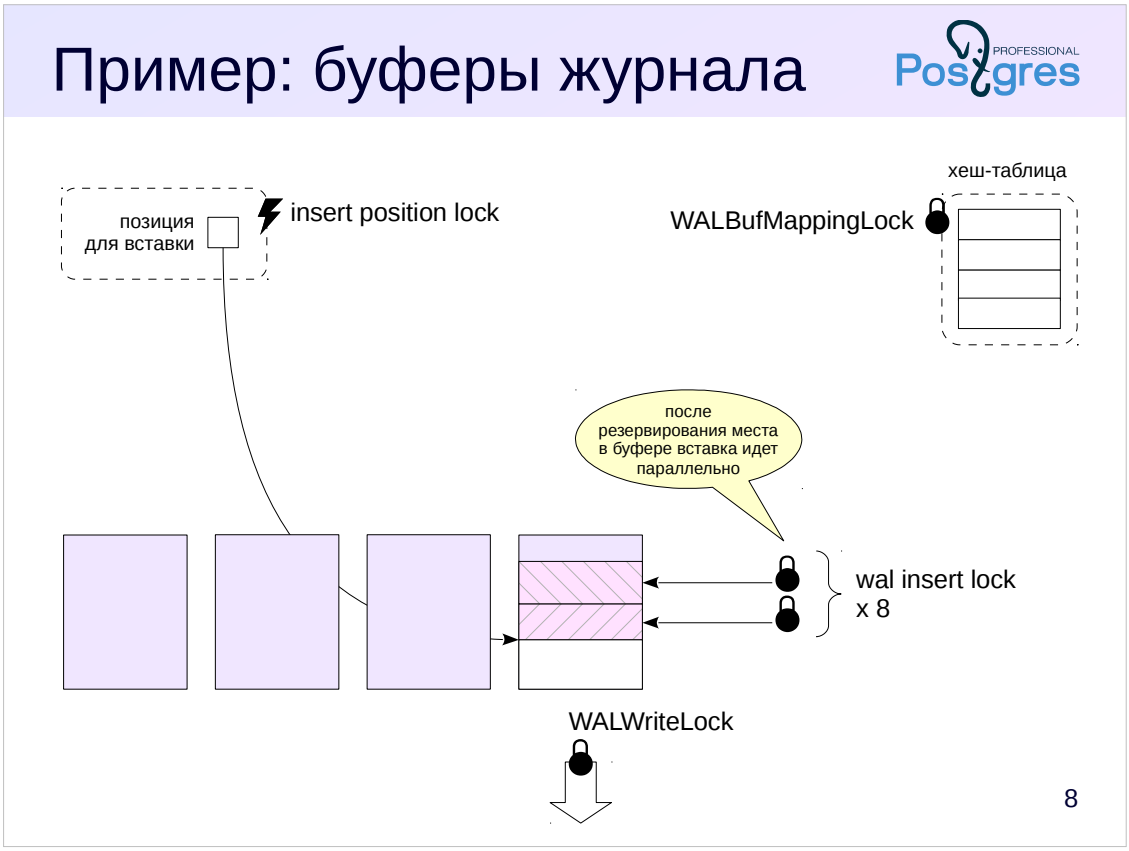
Чтобы обратиться к хеш-таблице, содержащей ссылки на буферы, процесс должен захватить легкую блокировку buffer mapping lock в разделяемом режиме, а если таблицу требуется изменять — то в исключительном режиме. Чтобы уменьшить гранулярность, эта блокировка устроена как *транш*, состоящий из 128 отдельных блокировок, каждая из которых защищает свою часть хеш-таблицы.

Доступ к заголовку буфера процесс получает с помощью спин-блокировки. Отдельные операции (такие, как увеличение счетчика) могут выполняться и без явных блокировок, с помощью атомарных инструкций процессора.

Чтобы прочитать содержимое буфера, требуется блокировка buffer content lock. Обычно она захватывается только для чтения указателей на версии строк, а дальше достаточно защиты, предоставляемой закреплением буфера. Для изменения содержимого буфера эта блокировка захватывается в исключительном режиме.

При чтении буфера с диска (или записи на диск) захватывается также 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 спин-блокировкой buffer strategy lock.



Еще один пример: буферы журнала.

Для журнального кэша тоже используется хеш-таблица, содержащая отображение страниц в буферы. В отличие от буферного кэша эта хеш-таблица защищена единственной легкой блокировкой WALBufMappingLock — поскольку размер журнального кэша меньше (обычно 1/32 от буферного кэша) и обращение к буферам более упорядочено.

Запись страниц на диск защищена легкой блокировкой WALWriteLock, чтобы только один процесс одновременно мог выполнять эту операцию.

Чтобы создать журнальную запись, процесс должен сначала зарезервировать место в странице. Для этого он захватывает спин-блокировку insert position lock. После того, как место зарезервировано, процесс копирует содержимое своей записи в отведенное место. Эта операция может выполняться несколькими процесами одновременно, для чего запись защищена *траншем* из 8 легких блокировок wal insert lock.

Здесь представлены не все блокировки, имеющие отношение к журналу предзаписи, но эта и предыдущая иллюстрации должны дать некоторое представление об использовании блокировок в оперативной памяти.

Мониторинг ожиданий

Ожидания в pg_stat_activity и семплинг

Типы ожиданий

Ожидания и семплинг

Когда процесс ожидает чего-либо,
этот факт отражается в представлении pg_stat_activity

- wait_event_type — тип ожидания
- wait_event — имя конкретного ожидания

Информация может быть не полна

- охвачены не все места в коде, в которых могут быть ожидания

Информация только на текущий момент

- единственный способ получить картину во времени — семплинг
- достоверная картина только при большом числе измерений

Для мониторинга ожиданий используется представление pg_stat_activity. Когда процесс (системный или обслуживающий) не может выполнять свою работу и ждет чего-либо, это ожидание можно увидеть в представлении. Столбец wait_event_type показывает тип ожидания, а столбец wait_event — имя конкретного ожидания.

Следует учитывать, что представление показывает только те ожидания, которые соответствующим образом обрабатываются в исходном коде. Если представление не показывает ожидание, это вообще говоря не означает со 100-процентной вероятностью, что процесс действительно ничего не ждет.

К сожалению, единственная доступная информация об ожиданиях — информация на текущий момент. Никакой накопленной статистики не ведется. Единственный способ получить картину ожиданий во времени — семплирование состояния представления с определенным интервалом. Встроенных средств для этого не предусмотрено, но можно использовать расширения, например, pg_wait_sampling.

При семплировании надо учитывать его вероятностный характер. Чтобы получить более или менее достоверную картину, число измерений должно быть достаточно высоко. Поэтому семплирование с низкой частотой не даст достоверной картины, а повышение частоты приводит к увеличению накладных расходов. По той же причине семплирование бесполезно для анализа короткоживущих сеансов.

Типы ожиданий

pg_stat_activity.wait_event_type

Блокировки

блокировки объектов	Lock
легкие блокировки	LWLock
закрепление буфера	BufferPin

Другие ожидания

ввод-вывод	IO
получение данных от другого процесса	IPC
получение данных от клиента	Client
активности в модуле расширения	Extension
«безделие»	Activity, Timeout

Все остальное — неучтенное время

Все ожидания можно разделить на несколько типов. Ожидания рассмотренных блокировок составляют большую категорию: ожидание блокировок объектов (значение Lock в столбце wait_event_type), ожидание легких блокировок (LWLock) и ожидание закрепленного буфера (BufferPin).

Но процессы могут ожидать и другие события. Ожидания ввода-вывода (IO) возникают, когда процессу требуется записать или прочитать данные. Процесс может ждать данные, необходимые для работы, от клиента (Client) или от другого процесса (IPC).

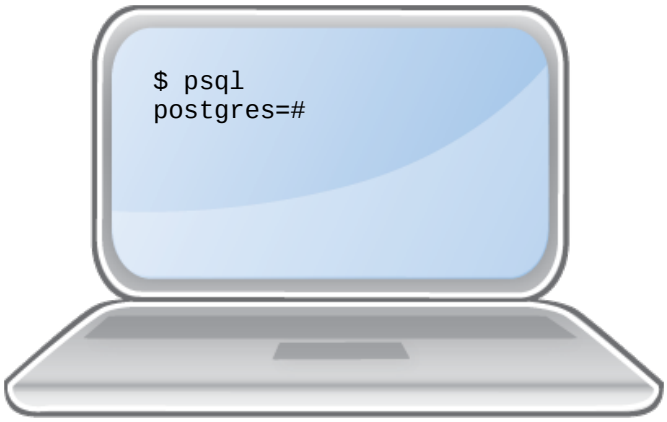
Расширения могут регистрировать свои специфические ожидания (Extension).

Бывают ситуации, когда процесс просто не выполняет полезной работы. К этой категории относится ожидание фоновых процессов в своем основном цикле (Activity), ожидание таймера (Timeout). Как правило, такие ожидания «нормальны» и не говорят о каких-либо проблемах.

Тип ожидания сопровождается именем конкретного ожидания: https://postgrespro.ru/docs/postgresql/10/monitoring-stats#WAIT-EVENT-TABLE

Если имя ожидания не определено, процесс не находится в состоянии ожидания. Такое время следует считать неучтенным, так как на самом деле неизвестно, что именно происходит в этот момент.

Демонстрация



В демонстрации мы используем файловую систему FUSE и проект slowfs, построенный с ее помощью.

Итоги

Блокировки в оперативной памяти реализуются по-разному

- спин-блокировки, легкие блокировки, закрепление буфера
- относительно короткое время и облегченная инфраструктура

Мониторинг текущих ожиданий с помощью представления pg_stat_activity

- семплинг для получения картины во времени

Практика

1. Открытый курсор удерживает закрепление буфера, чтобы чтение следующей строки выполнялось быстрее. Убедитесь в этом с помощью расширения pg_buffercache.
2. Откройте курсор по таблице и, не закрывая его, выполните очистку таблицы (VACUUM). Будет ли очистка ждать освобождения закрепления буфера?
3. Повторите эксперимент, выполнив очистку с заморозкой (VACUUM FREEZE). Убедитесь, что в профиль ожиданий обслуживающего процесса попало ожидание закрепления буфера.

1. Расширение pg_buffercache было рассмотрено в модуле «Журнал», тема «Буферный кэш». Представление pg_buffercache содержит столбец pinning_backends, который показывает количество процессов, закрепивших этот буфер. Нужный буфер можно найти по условию relfilenode = pg_relation_filenode(*имя_таблицы*).

2. Для проверки удобно воспользоваться вариантом команды VACUUM VERBOSE.