УДК 004

ПОДХОДЫ К СБОРУ ОПЕРАТИВНОЙ СТАТИСТИКИ СЕССИЙ POSTGRESQL

Соколова Е.В., студент гр. 227, VI курс
Научный руководитель: Гайсарян С.С., к.ф.-м.н., профессор
Московский государственный университет
имени М.В. Ломоносова
г. Москва

Системы управления базами данных в современном мире являются необходимым инструментом в IT сфере. Проведение банковских транзакций, хранение данных приложения, извлечение информации для аналитических выкладок и даже оплата проезда в общественном транспорте — всё это трудно представить без грамотно организованного взаимодействия с базой данных.

PostgreSQL — объектно-реляционная система управления базами данных. Согласно рейтингу DB-Engines, PostgreSQL является четвёртой по популярности СУБД во всём мире [1]. PostgreSQL обеспечивает очень хорошую поддержку стандарта языка SQL и также предоставляет интересные и практически полезные дополнительные возможности. Одним из главных достоинств PostgreSQL является расширяемость — возможность пользователя вносить свои типы данных, функции, методы доступа и т.д. Кроме того, PostgreSQL — свободно распространяемый продукт с открытым исходным кодом, который доступен на большом числе платформ [2].

Требования к СУБД растут с каждым днём. Одно из ключевых направлений — ускорение взаимодействия с базой данных. В интернете ежедневно появляется множество статей о корректных настройках и использовании различных инструментов, позволяющих администраторам баз данных (DBA) делать выводы о необходимости принятия тех или иных решений.

Одним из самых популярных инструментов DBA является статистика сессий. Система накопительной статистики в PostgreSQL представляет собой подсистему, которая собирает и отображает информацию о работе сервера. В настоящее время в ней отслеживаются обращения к таблицам и индексам как на уровне блоков на диске, так и на уровне отдельных строк. Также подсчитывается общее число строк в каждой таблице, и собирается информация о выполняемых для каждой таблицы действиях очистки и анализа [3]. В процессе выполнения запроса к БД происходит «накопление» статистических параметров. Обновление имеющихся значений статистик с учётом «накопленных» произойдёт только после завершения запроса (или транзакции — последовательности команд, объединённых в одну атомарную операцию). Таким образом, функция, являющаяся совокупностью всех существующих в PostgreSQL

методов получения какой-либо доступной в момент времени t статистики для сессии s является дискретной.

Указанный подход ко сбору показателей в некоторых ситуациях может вводить администратора в заблуждение [4]. Например, СУБД в автоматическом режиме запустила в одной из сессий аналитический запрос, на выполнение которого уйдёт больше суток и ощутимое количество ресурсов. Администратор видит падение производительности, но не может вычислить причину — статистика сессий находится в рамках привычных значений. По прошествии времени, после завершения «долгого» аналитического запроса, супервизор увидит резкий скачок значений, собираемых системой накопительных статистик, для бэкенда, в котором он был запущен. Вследствие этого, остаётся актуальным вопрос о создании инструмента, позволяющего собирать статистику оперативно, т.е. в момент выполнения запросов и транзакций.

Идейным вдохновителем оперативных статистик можно считать Active Session History (ASH) в СУБД Oracle. Это набор статистик по отдельным сессиям/транзакциям с момента их старта, пополняемый новыми записями спустя установленный промежуток времени [5]. Принципиально понимать, что в Oracle статистические значения обновляются «в прямом эфире», т.е. в момент выполнения операций. Поэтому ASH лишь хранит «снимки» значений параметров, накапливая их, чтобы видеть изменения в течение времени. Сейчас, в PostgreSQL 16.2 и более ранних версиях, нет механизма по оперативному получению статистики, поэтому для реализации «постгрессового аналога ASH», потребуется принципиально новый относительно существующих в PostgreSQL подход к получению статистических сведений из бэкенда, выполняющего SQL-запрос.

Требуется разработать алгоритм сбора доступных в бэкенде статистик всей сессии, текущей транзакции и текущего statement-а (команды). В первую очередь, особый интерес представляют статистики, собираемые модулем pg_stat_statements [6], поэтому для начала ограничимся получением только этих показателей. В дальнейшем можно рассмотреть возможность добавления и других параметров. Для хранения получаемых актуальных значений создать хэш-таблицу в разделяемой памяти (её размер заранее известен, поскольку количество собираемых показателей мы ограничили набором из pg_stat_statements, а количество записей будет равно max_connections — системной константе, ограничивающей максимальное количество подключений к базе, т.е. являющейся верхней границей возможного количества сессий).

Рассмотрим возможные подходы к сбору оперативных статистик сессий. Для этого изучим вопрос получения каких-либо переменных из активного бэкенда «на лету», т.е. в процессе выполнения пользовательских или системных операций.

В первую очередь рассмотрим возможность прерывания по сигналу. Этот метод используется расширением pg_query_state, реализующим в PostgreSQL одноимённую функцию pg_query_state(pid), позволяющую увидеть

план запроса, выполняющегося в бэкенде с идентификатором pid. Системная команда EXPLAIN ANALYSE query может отобразить план с подробной статистической информацией, собранной по каждому узлу плана query, но для получения этих значений запрос требуется запустить на выполнение и дождаться его завершения [7]. Модуль pg_query_state добавляет функционал прерывания по пользовательскому сигналу. В качестве обработки отправленного сигнала бэкенд посылает посредством сообщений результат команды EXPLAIN ANALYSE с собранными к текущему моменту значениями параметров (при повторном вызове функции pg_query_state для одного и того же запроса можно проследить увеличение собираемых показателей).

К плюсам описанной реализации можно отнести возможность повсеместного использования пользовательских сигналов и переиспользование части кода расширения pg_query_state. К минусам - повышенные накладные расходы. «Опросив» бэкенд функцией pg_query_state без сбора статистики по времени и буферам в процессе выполнения операций MergeJoin + GroupAggregate мы получим увеличение времени выполнения на 3,27% относительно запуска этой же команды без прерывания. При включении информации по времени и/или используемым буферам, погрешность ещё выше. Для операции Cross join без получения показателей времени и буферов накладные расходы составят 2,31% увеличения длительности выполнения. Такая погрешность допустима при единовременном опросе интересующей нас операции, поскольку в реальных условиях нас заинтересует план запроса со статистикой по выполнению только тогда, когда он тратит ощутимое количество ресурсов системы (например, долго выполняется). При применении указанного подхода для реализации оперативных статистик сессий, прерывания по сигналу будут не в одном бэкенде, а во всех активных бэкендах (что может дополнительно создать гонку и риск взаимных блокировок). Значит, обозначенные накладные расходы будут кратно выше, что недопустимо для инструмента, который планируется запускать для опроса состояния БД по таймеру (не реже раза в минуту).

В качестве альтернативы можно использовать подход с hook-ами. Hook — это интерфейс для взаимодействия с внешними системами или ресурсами, позволяющий «перехватить» функцию и изменить её поведение. К плюсам можно отнести минимизацию изменений в существующем программном коде (только добавление нового функционала, без изменения поведения существующего), простоту реализации. Hook-и потребуют дополнительных проверок состояния и накладных расходов больше, чем у сигналов. При создании модуля pg_query_state рассматривался вариант прерывания посредством hook-а, он давал просадку по времени 0,99% на запросе MergeJoin + GroupAggregate (против 3,27% у сигналов), но для операции Cross join 14,58% (против 2,31%). Результат не только медленнее версии с прерываниями по пользовательскому сигналу, но и может давать недопустимое для реальной системы замедление операции (более 10%).

В качестве третьего варианта возможной реализации рассмотрим механизм отложенных прерываний CHECK_FOR_INTERRUPTS. Данный подход позволяет проверить, были ли получены требования прерваться, пока процесс спал. Таким методом реализован сбор статистик вакуума в процессе его выполнения в модуле pgpro_stats. Такой подход получения данных схож с первым методом (прерыванию по сигналу), при этом позволяет при необходимости отложить обработку прерывания, что поможет снизить накладные расходы на выполнение и избежать излишних гонок состояний. Поскольку основной критерий выбора инструмента мониторинга СУБД — отсутствие излишней нагрузки на сервер, в качестве способа реализации прототипа оперативных статистик сессий был выбран метод, использующий механизм отложенных прерываний.

Подходы к получению оперативных статистик обсуждались с коллегами из компании Postgres Professional, занимающей второе место в мировом топе контрибьюторов открытой СУБД PostgreSQL [8], российскими разработчиками системы управления базами данных Postgres Pro на основе PostgreSQL [9]. В процессе дискуссии было решено разместить основной функционал, касающийся обработки полученных параметров, в коде модуля pg_stat_statements. Во-первых, поскольку это расширение, а не код ядра, использование нового функционала опционально, т.е. не включено по умолчанию, а управляется пользователем или администратором базы данных. Во-вторых, это «ванильное», т.е. входящее в дистрибутив PostgreSQL расширение, сосредоточенное на сборе статистики. Значит, впоследствии реализацию оперативных статистик можно будет предложить сообществу и, после внесения правок по рекомендациям других контрибьюторов, внести в open source версию PostgreSQL. Более того, это самое «логичное» место расположения нововведений с точки зрения расположения других функций, отвечающих за разного рода статистики (из решений с открытым исходным кодом).

Оперативные статистики сессий — горячо желаемый администраторами баз данных инструмент. На настоящий момент существует большое количество open source решений, старающихся реализовать сбор статистик имеющимися инструментами, но по причине отсутствия в PostgreSQL механизмов, опрашивающих запросы в момент их выполнения, по-настоящему «оперативного» получения статистик сейчас не реализовано. Рассматриваемый инструмент требует фундаментального подхода и создания новых механизмов в СУБД PostgreSQL. Созданный прототип основывается на механизме отложенных прерываний. Реализованы оперативный сбор параметров, отслеживаемых модулем pg_stat_statements (в нём же и расположен основной функционал нового инструмента), запись в хэш-таблицу в разделяемой памяти и доступ к ней посредством представления (VIEW).

Список литературы:

1. DB-Engines Ranking (March 2024) [Электронный ресурс] // DB-Engines.com URL: https://db-engines.com/en/ranking (дата обращения: 31.03.2024)

2. Моргунов Е. П., под ред. Рогова Е. В., Лузанова П. В. PostgreSQL. Основы языка SQL //учеб. пособие/ЕП Моргунов. – 2018.

3. Документация PostgreSQL 15. Система накопительной статистики [Электронный ресурс] // PostgresPro.ru URL: https://postgrespro.ru/docs/postgresql/15/monitoring-stats (дата обращения: 31.03.2024)

4. Колосков В. Мониторинг PostgreSQL. Новые возможности анализа производительности 1С и других систем. Часть 1: счётчики [Электронный ресурс] // Habr.com URL: https://habr.com/ru/companies/softpoint/articles/747322/ (дата обращения: 31.03.2024)

5. Dias K. et al. Automatic Performance Diagnosis and Tuning in Oracle //CIdR. – 2005. – С. 84-94.

6. Kozhevnikov V. A., Sabinin O. Y., Shats J. E. DEVELOPMENT OF CROSS-PLATFORM UTILITY FOR THE DBA USING A LIBRARY FOR WORKING WITH BOTS //Theoretical & Applied Science. – 2018. – №. 6. – С. 127-131.

7. Исходный код расширения pg_query_state. README-файл [Электронный ресурс] // GitHub.com URL: https://github.com/postgrespro/pg_query_state/blob/master/README.md (дата обращения: 31.03.2024)

8. Conway S. The Importance of Giving Back: Analyzing Code Contributions to PostgreSQL 15 [Электронный ресурс] // EnterpriseDB.com URL: https://www.enterprisedb.com/blog/importance-of-giving-back-to-postgresql (дата обращения: 31.03.2024)

9. Ванина М. Ф., Ерохин А. Г., Фролова Е. А. К вопросу о поиске альтернативных отечественных решений при обучении технологиям работы с базами данных в высших учебных заведениях //Методические вопросы преподавания инфокоммуникаций в высшей школе. – 2018. – Т. 7. – №. 2. – С. 15-18.