СЛАЙД 1

Здравствуйте, уважаемая комиссия.

(Меня зовут …)

(Тема моей выпускной квалификационной работы звучит следующим образом …)

СЛАЙД 2

Целью выпускной квалификационной работы является разработка метода параллельного выполнения запросов к СУБД PostgreSQL в пределах одного соединения.

СЛАЙД 3

Согласно исследованию, операция доступа к базе данных объемом 100.000 записей в случае многопоточной программы выполняется в 1000 раз быстрее. Однако программа завершается с ошибкой нехватки памяти при десяти тысячах и более соединений.

Ошибка связана с необходимостью создания соединения в каждом потоке. В то время, как операция соединения одна из самых дорогостоящих.

СЛАЙД 4

Причины высоких затрат кроются в архитектуре СУБД. ***(Дорогая, потому что такая архитектура)***

Данная работа является модификацией PostgreSQL 14-ой версии.

В архитектуре Postgres-а выделяют 3 основные подсистемы.

Первая – клиентская часть. Состоит из пользовательского приложения и библиотеки libpq. Библиотека содержит набор функций для создания соединений.

Вторая – серверная. Соединение на сервере принимается процессом-демоном *postmaster*, который в дальнейшем с помощью системного вызова *fork()* создаст новый процесс Postgres для обслуживания данного соединения, а также ряд служебных процессов.

Третья часть сформирована из хранилища данных и средств его управления.

СЛАЙД 5

Один из подходов сокращения числа подключений – использование пула соединений.

Пул представляет из себя набор заранее открытых и готовых к использованию соединений с БД.

И хотя в PostgreSQL отсутствует встроенный пул соединений, он может быть реализован на основе библиотеки libpq или в качестве внешней службы.

СЛАЙД 6

Преимущество использования пула – это увеличение пропускной способности транзакции до 60%.

Однако каждая из представленных реализаций имеет свои ограничения. Так, главным недостатком пула на основе библиотеки – затраты на его разработку. В том числе необходимость изменения архитектуры приложения. Недостаток пула в качестве внешней службы – однопоточная реализация самой службы. Встроенный же пул доступен только в коммерческой версии.

СЛАЙД 7

Разрабатываемый метод состоит из двух этапов обработки запроса: 1ый – отправка запроса серверу, включающий а) формирование очереди запросов и б) отправку запроса из очереди; 2й – получение ответа от сервера.

СЛАЙД 8

Работа функции отправки запроса серверу начинается с блокировки параметров соединения. Блокировка необходима для защиты параметров от изменений другими потоками, т.к. результат выполнения напрямую зависит от параметров.

После того как была произведена блокировка, выполняется проверка корректности самих параметров и инициализация соответствующих полей. В том числе, определяется точка входа очереди команд. После указания протокола запросов, команда может быть добавлена в очередь.

СЛАЙД 9

Ожидание поступления результата от сервера реализовано с помощью бесконечного цикла, на каждой итерации которого выполняется проверка состояния сервера. Пока сервер готов вернуть данные, выполняется их чтение и запись во входной буфер. Как только состояние сервера меняется на «свободен», выполняется обработка полученных данных. Обработка включает в себя чтение типа сообщения, его длины, и непосредственно сообщение результата.

Перед выходом из функции выполняется проверка корректности полученных данных и разблокировка параметров соединения для обработки следующего запроса.

СЛАЙД 10

Помимо встроенного модуля, был реализован внешний, выполняющий вызов разработанного метода. Кроме того, пользователю доступны однопоточная, многопоточная реализации, реализация с внешним пулом и сравнение времени выполнения каждой.

Внешний пул был разработан с использованием умных указателей для предотвращения возможной утечки ресурсов. Сам пул был реализован в качестве очереди соединений.

СЛАЙД 11

В данной работе было проведено исследование времени работы реализованного метода. Выполнялось сравнение времени выполнения простого запроса для 4 реализаций.

Согласно результатам, наибольшее время выполнения было зафиксировано у однопоточной реализации (0.42 секунды при 500 потоках). У многопоточной реализации наибольшее преимущество перед однопоточной достигается в случае 500 соединений (в 1.7 раз работает быстрее), однако при малом числе (10) соединений результаты работы отличаются незначительно (в 0.99 раз).

Из опыта видно, что конкурентоспособность по времени разработанному методу составляет только пул соединений.

СЛАЙД 12

Поэтому было проведено другое исследование, где выполнялось сравнение времени работы пула, использующего библиотеку libpq и пула, реализованного в качестве внешней службы (PGBouncer), с разработанным методом.

Согласно результатам эксперимента, при более 100 соединений PGBouncer теряет временную эффективность: в 2 раза работает дольше, чем внешний пул. Однако при 10 соединениях работает быстрее разработанного метода в 1.24 раза.

СЛАЙД 13

Также было выполнено исследование требуемых ресурсов. Был проведен эксперимент, фиксирующий затраты памяти для каждой реализации в случае создания 10 соединений и выполнения простого запроса.

Согласно результатам, наибольшее потребление памяти (593 байта) у многопоточной реализации, что в 3 раза больше, чем память, потребляемая разработанным методом. Также, метод требует в 3 раза меньше памяти, чем внешний пул.