НОВЫЕ ПОДХОДЫ ОПТИМИЗАЦИИ ЗАПРОСОВ В БАЗАХ ДАННЫХ

Гобрик О.Д., Гуринович А.Б.
Кафедра информационных технологий автоматизированных систем, Кафедра вычислительных методов и программирования, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: oleg.gobrik@gmail.com, gurinovich@bsuir.by

В работе исследуется проблема обеспечения эффективности функционирования систем управления базами данных (СУБД) с помощью оптимизации запросов. Показываются наиболее распространенные виды параллельных реляционных СУБД. Причиной наибольших ошибок оптимизатора запросов является оценка стоимости запроса.

Введение

Рост многообразия моделей данных и способов их представления на физических носителях (т.е. типов БД), и многообразия средств взаимодействия с БД (т.е. СУБД) приводит к появлению новых методов решения оптимизации запросов. Отметим, что ядром БД является модель данных, т.е. совокупность структур данных и операций их обработки. Кроме того, СУБД по своему предназначению делятся на операционные и предназначенные для работы с хранилищами данных, содержащими очень большой объем информации, подготовка представления которой занимает значительное время. Несмотря на растущую сложность реализации и большое обилие алгоритмов, обеспечивающих большую производительность, по-прежнему существует некоторое число проблем, связанных с производительностью запросов. В исследовании планируется описать подходы оптимизации запросов баз данных.

I. Подходы оптимизации запросов

На данный момент существуют 3 направления исследований.

1. Построение параллельных вычислений на основе использования графических процессоров (GPU). В состав обычного CPU входят от 1 до 32 ядер, а в состав GPU – несколько тысяч простых процессоров, например, GeForce RTX 2080 обладает 2944 ядрами CUDA. Под «простыми» понимаются процессоры, не содержащие компонент проверки условий, кэш и т.д. Имея функцию, выполняющую некоторую работу, разработчик назначает ее исполнение на тот или иной GPU. Модель программирования GPU описывает 6 типов областей памяти, различных по скорости доступа, прав на запись-чтение, объему. Наличие большого числа процессоров и быстрой памяти обеспечивает эффективность выборки данных со сложным условием, так как каждый поток исполняется независимо от других, оперируя только с малой частью обрабатываемых данных. В результате существенно снижается различие между временем загрузки данных с медленного носителя в оперативную память и временем, затрачиваемым на исполнения запроса и возвращения результата[3].

2. Разработка моделей и методов обработки и хранения больших данных. Под термином «большие данные» понимают электронные данные, характеризующиеся большим объемом, разнообразием и скоростью, с которой структурированные и неструктурированные данные поступают по сетям передачи в процессоры и хранилища, а также наличием эффективных процессов переработки данных в требуемую информацию. Именно актуальность разработки средств эффективной обработки больших данных привела к появлению концепции NoSQL[4]. Ее суть состоит в проектировании архитектуры, обладающей свойством адаптации к возрастающим объемам данных. При этом сохранение эффективности процессов обработки данных обеспечивается за счет высокой пропускной способности и потенциально неограниченного горизонтального масштабирования.

Также для обработки данных применяют облачные технологии. Для оптимизации в облаке используется 2 способа. Первый способ реализует параллельную обработку на уровне запросов. В этом случае алгебраическое выражение запроса разбивается на подвыражения, которые могут выполняться одновременно на разных процессорах или ядрах одного многоядерного процессора. Второй способ основан на параллельной реализации операций, которые составляют запрос. Далее отдельно рассмотрен способ оптимизации путем использования машинного обучения.

II. Применение алгоритмов машинного обучения

Для оптимизации запросов в реляционных СУБД применяется стоимостной оптимизатор. При стоимостной оптимизации оптимальность плана запроса определяется значением, которое именуется стоимостью плана. Данное значение рассчитывается как совокупная стоимость каждой операции в запросе и определяется в зависимости от планируемого числа записей, которые

будут извлечены из базы (значение cardinality). Для решения проблемы производительности при взаимодействии с данными, которые имеют зависимость между собой, применяются алгоритмы, позволяющие подбирать наилучший план. Данная реализация называется "Adaptive Query Optimization". Она позволяет выбирать наиболее производительный план при повторном выполнении запросов одного типа, обучаясь после каждого запроса[5].

Таким образом, решается проблема актуальности статистических данных для осуществления расчётов системой машинного обучения. Теперь, произведя всю цепочку описанных действий, можно получить оценку стоимости каждой вершины плана и выбрать наиболее оптимальную с точки зрения объёмов затрачиваемых на выполнение последней ресурсов. Технологии машинного обучения позволят автоматизировать и оптимизировать работу с формализованными данными, собирая при этом актуальную статистику и самосовершенствуясь.

Не менее важным вопросом, является предоставление наиболее точных исходных данных. Оптимального размещения файлов реляционной БД по узлам компьютерной сети требует ряда информационных массивов исходных данных, значительная часть которых может быть получена лишь в усредненном или приниженном виде. Это такие характеристики, как интенсивности запросов, время пересылки и обработки запросов, объёмы запросов и ответов на запросы. Точность собранной статической информации будет решающим образом влиять на конечный результат реализации выбранной математической модели и, следовательно, на производительность системы, работающей с реляционной БД[6].

В PostgreSQL существует реализация адаптивной оптимизации. Здесь используется аналогичный метод подбора значения cardinality для схожих запросов. В качестве алгоритма используется модифицированный метод KNN (K-ближайших соседей), способный обрабатывать ситуацию изменения данных в таблице.

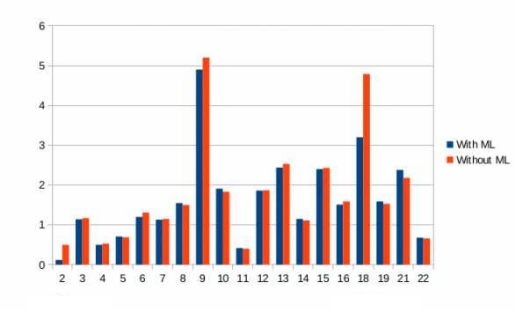

Рис. 1 – Диаграмма скорости выполнения запросов

Хотя данная реализация способна справляться с различными изменениями в данных, все же рекомендуется реинициализировать алгоритм после внесения обширных изменений в данные (в контексте KNN это означает удаление созданных объектов с информацией о cardinality). Данный подход ускоряет время работы СУБД. Для тестирования использовался бенчмарк TPC-H. На одном из типов запросов бенчмарка ускорение составляет 30-45% (2-ой и 18-ый запросы).

Тем не менее, данная реализация имеет несколько недостатков. В качестве признаков здесь используются значения селективности (процент кортежей, удовлетворяющих конкретному условию), извлекаемые из статистики, что может работать не совсем корректно, так как неэквивалентные условия в запросе могут превращаться в эквивалентные по значению результаты предсказания. Так же существует проблема с работой данного плагина на репликах, поскольку параметры алгоритма хранятся в таблице в базе.

В условиях стремительно растущей потребности в эффективных способах хранения и манипуляций с большими объёмами данных поиск новых решений является необходимым.

III. Заключение

Существующие подходы к проблеме оптимизации запросов обладают рядом недостатков, для решения которых предлагается 3 метода: построение параллельных вычислений на основе использования графических процессоров, разработка моделей и методов обработки и хранения больших данных, и применение алгоритмов машинного обучения. Установлено, что использование машинного обучения ведет к улучшению работы планировщика и, соответственно, ускорению работы СУБД, а также отмечены проблемы в алгоритме оценки выборочности. В дальнейшем, при использовании более сложных алгоритмов машинного обучения, возможно ускорить обработку запросов и работу СУБД в целом.

Список литературы

1. А. Ю. Васильев, Работа с PostgreSQL: настройка и масштабирование, 2015 - 203 с.
2. http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1638/12-Shchetinin.pdf?sequence=1
3. https://www.academia.edu/8903054/
4. Медетов А.А. Термин Big Data и способы его применения // Молодой ученый, 2016.207-210
5. http://www.libeldoc.bsuir.by/bitstream/123456789/32182/1/Sharayev_Optimizatsiya.pdf
6. Маркин С.Д., Головинов А.О. Интеграция методов машинного обучения в планировщик SQL-запросов // Научное сообщество студентов XXI столетия. Технические науки: сб. ст. по мат. LIII междунар. студ. науч.-практ. конф. № 5(52).