КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2015 Т. 7 № 3 С. 649–655

Ки&М

СЕКЦИОННЫЕ ДОКЛАДЫ

УДК: 004.65

Оптимизация запросов в распределенных базах данных и распространение технологии, (облачных вычислений)

А. В. Богданов, Тхурейн Киав Лвин[a]

Санкт-Петербургский государственный университет,
Россия, 198504, г. Санкт-Петербург, Петергоф, Университетский просп., д. 35

E-mail: [a]trkl.mm@mail.ru

Получено 21 января 2015 г.

Оптимизация это сердце для реляционных СУБД. Она анализирует SQL заявления и определяет наиболее эффективный план доступа для удовлетворения каждого запроса. Оптимизация решает эту задачу и анализирует SQL заявления определяя, какие таблицы и столбцы должны быть доступны. Затем запросы информационной системы и статистические данные, хранящиеся в системном каталоге, определяют наилучший метод решения задач, необходимых для удовлетворения этой просьбы.

Ключевые слова: оптимизация запросов, облачные вычисления, распределенные базы данных

Query optimization in relational database systems and cloud computing technology

A. V. Bogdanov, Thurein Kyaw Lwin

Saint Petersburg State University, University ave. 35, Peterhof, St. Petersburg, 198504, Russia

Optimization is the heart of relational Database Management System (DMBS). Its can analyzes the SQL statements and determines the most efficient access plan to satisfy every query request. Optimization can solves this problem and analyzes SQL statements specifying which tables and columns are available. And then request the information system and statistical data stored in the system directory, to determine the best method of solving the tasks required to comply with the query requests.

Keywords: query optimization, cloud computing technology, relational database systems

Citation: *Computer Research and Modeling*, 2015, vol. 7, no. 3, pp. 649–655 (Russian).

Оптимизация — это сердце для реляционных СУБД. Она анализирует SQL-заявления и определяет наиболее эффективный план доступа для удовлетворения каждого запроса. Оптимизация решает эту задачу и анализирует SQL-заявления, определяя, какие таблицы и столбцы должны быть доступны. Затем запросы информационной системы и статистические данные, хранящиеся в системном каталоге, определяют наилучший метод решения задач, необходимых для удовлетворения этой просьбы.

Оптимизация существенна и для экспертной системы при доступе к базе данных. Экспертная система представляет собой набор стандартных правил, который в сочетании с ситуационными данными может либо выдать рекомендацию, либо вернуться к мнению экспертов. Реляционный оптимизатор показывает влияние мнения экспертов о методах поиска данных. Понятие «оптимизация» для доступа к данным в СУБД дает очень мощный потенциал, который мы сегодня воспринимаем как должное. На сегодняшний день доступ к реляционным данным достигается путем запроса СУБД. Независимо от того, какие данные физически хранятся и обрабатываются, SQL могут быть использованы для доступа к данным. Такое разделение критериев доступа, физических характеристик и типа систем хранения называется физической независимостью данных, и оптимизация имеет решающее значение в достижении этой физической независимости.

Если индексы в таблице удаляются, вы все равно можете получить доступ к данным. Если столбец будет добавлен в таблицу, к которой осуществляется доступ, с данными все еще можно манипулировать без изменения программного кода. Все это возможно потому, что физические пути доступа к данным не заложены программистами в прикладной программе, а создаются оптимизатором.

Оптимизация выполняет сложные расчеты, основанные на множественной информации. Для упрощения функциональности оптимизатора мы можем разложить его деятельность на четыре этапа:

получение и проверка синтаксиса SQL-заявления;

анализ окружающей операционной среды и оптимизация методов удовлетворения SQL-заявления;

создание команды для выполнения оптимизированных SQL;

выполнение инструкции или сохранение ее для будущего исполнения.

Оптимизация SQL имеет множество стратегий. Производители СУБД не публикуют углубленные детали внутренней работы своих оптимизаторов, но хороший оптимизатор должен будет минимизировать потребные ресурсы (cost-based). Это означает, что оптимизатор всегда будет пытаться сформулировать путь доступа для каждого запроса, чтобы уменьшить общую стоимость транзакций. Чтобы достичь этого, оптимизатор запросов будет применять формулы, которые оценивают стоимость и вес многих факторов для каждого потенциального пути доступа, такие как стоимость процессора, I/O-стоимость, статистическая информация в системном каталоге, а также фактическое заявление SQL.

Без статистики, которая хранится в системном каталоге, оптимизация будет трудно осуществима. Эта статистика оптимизатора с информацией о состоянии таблиц, которые будут доступны в заявлении SQL, и данные о том, что в настоящее время уже оптимизировано. Типы статистической информации включают в себя:

информацию в таблицах, включающую общее число строк, информацию о компрессии и общее количество страниц;

информацию в столбцах, в том числе количество дискретных значений для столбца, и распределение диапазона значений, хранимых в столбце;

информацию в табличных пространствах, в том числе количество активных страниц;

текущее состояние индекса, включается ли индекс, существует или нет организация индекса (number of leaf pages and number of levels), число дискретных значений ключа индекса и делится ли индекс по группам;

информация табличного пространства и табуляторы.

Статистические данные собираются и хранятся в системном каталоге, и поэтому полезно выполнить операции обновления статистики (например, RUNSTATS или UPDATE STATISTICS). Для более эффективной работы с DBA важно контролировать, чтобы статистические данные накапливались в процессе выполнения транзакций, особенно в производственной среде.

В настоящее время происходит развитие и распространение технологии «облачных вычислений» [Никульчев, 2003; Плужник, Функционирование…, 2013]. Растущий спрос на услуги провайдеров, предлагающих широкий спектр услуг в области облачных вычислений для большого числа пользователей по всему миру, приводит к увеличению количества приложений, целью которых является обработка больших массивов данных. Функционирование баз данных в облачной среде приводит к необходимости поиска новых инструментов [Плужник, Слабоструктурированные…, 2013].

Оптимизация запросов в облачных БД SQL

Обработка запроса сводится к преобразованию высокоуровневого запроса в эквивалентную низкоуровневую форму, и основной трудностью при этом является обеспечение эффективности преобразования с учетом специфики облачных хранилищ. Стандартные SQL-запросы используют соединения (join), выборку (select), проекции (projections), группировки (group-by). В [Zhang, 2012] приводится описание архитектуры, предназначенной для обработки и хранения больших объемов данных на основе семантических алгоритмов поиска плана исполнения текущего запроса в глобальной схеме исполнения запросов (рис. 1).

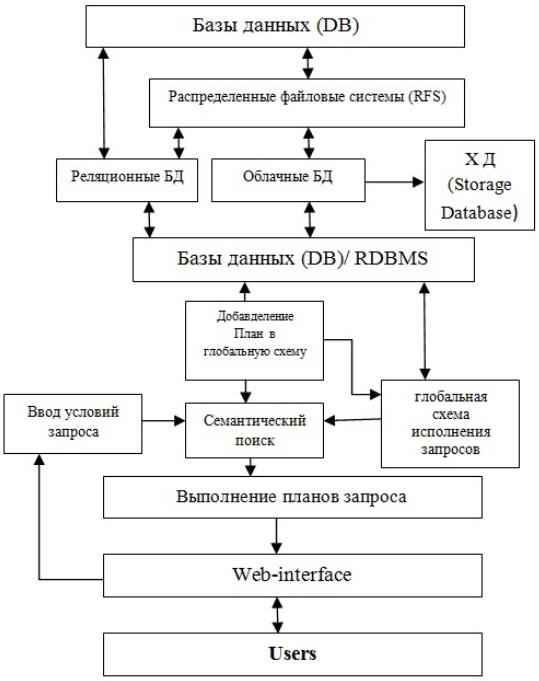

Рис. 1. Архитектура системы, основанной на SQL-запросах

Ключевые принципы указанной архитектуры состоят в следующем.

1. Все файлы хранятся в локальной файловой системе (например, файловая система Windows, Linux и т. д.).

2. Облачная БД предназначена для хранения и управления огромными массивами файлов индекса и метаданных. При этом следует отметить, что облачная база данных со всем содержимым развернута поверх распределенной файловой системы.

3. Ввод запросов и получение результатов выполняется посредством пользовательского веб-интерфейса.

4. После получения пользовательского запроса выполняется семантический поиск плана исполнения текущего запроса в глобальной схеме (как подмножество).

Архитектурно зависимые решения

Основной интерес указанного подхода заключается в том, что его целью является балансировка нагрузки на оборудование посредством миграции виртуальных машин в облачном окружении, что опосредованно приводит к повышению качества поиска. Зачастую для управления ресурсами внутри крупномасштабных центров обработки данных разрабатываются и внедряются централизованные решения, однако в этом случае возникновение сбоя на управляющем узле приводит к неработоспособности всей системы в целом.

Как показано на рис. 2, каждый активный узел в процессе функционирования выборочно с заданным интервалом отправляет собственный индекс загруженности некоторым узлам системы, в то же время получая индексы загруженности случайно выбранных активных узлов. При этом целевые узлы меняются на каждой итерации. Информация о загруженности других узлов добавляется в вектор загруженности текущего. Таким образом, средняя длина вектора загруженности узла равна количеству итераций отправки индекса. Информация о загруженности будет храниться децентрализовано, что позволит избежать неприятностей в случае выхода из строя части узлов, еще одним положительным моментом является то, что сетевой трафик будет распределен по всем активным узлам.

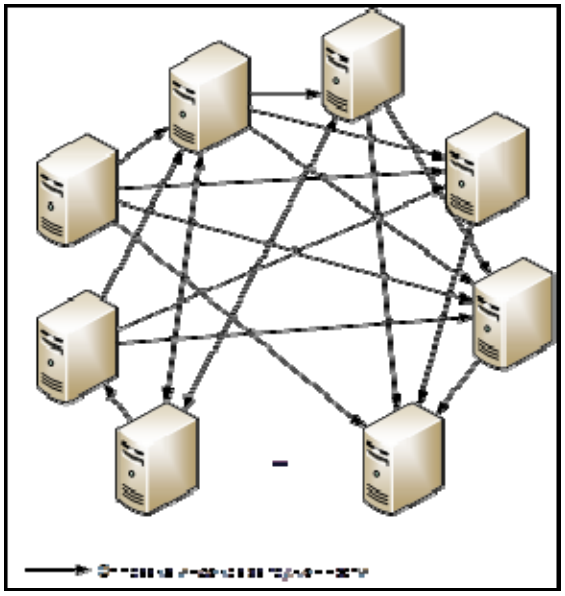

Рис. 2. Децентрализованный обмен индексом загруженности

Поскольку виртуальные машины служат хостом для развертывания разнообразных приложений с различающимися рабочими нагрузками на ЦПУ, то со временем загруженность физических ЦПУ может сильно меняться. При этом решение о миграции виртуальной машины может быть принято в двух случаях:

1) когда использование ЦПУ превышает определенный уровень (верхний порог); целью установления верхнего порога является сохранение дополнительных вычислительных мощностей на случай возникновения ситуаций с резким (незапланированным) повышением нагрузки;

2) когда использование ЦПУ ниже определенного уровня (нижний порог), узел используется недостаточно; цель установления нижнего порога состоит в том, чтобы по возможности

большее число физических узлов было переведено в «спящий» режим, что позволит снизить энергопотребление.

После того как принято решение о миграции виртуальной машины, стартует поиск узла назначения [Wang, 2013]. Для этого выполняется обход вектора загруженности текущего узла с целью обнаружения узла с наименьшим потреблением ЦПУ при условии попадания в заданные интервалы. Если такой узел обнаружить не удается, выполняется поиск такого узла, индекс загруженности которого при переносе на него выбранной ВМ не превышает нижней границы загруженности. Если же и в этом случае поиск не дает результатов, один из узлов, находящихся в «спящем» режиме, переводится в активное состояние и выполняется миграция.

Недостатки рассматриваемого подхода

1. Описываемые в 10 алгоритмы (псевдокоды) не гарантируют обязательности выбора ВМ для миграции даже при условии необходимости в этом и наличии свободных физических узлов.
2. Не рассматривается оптимальность выбора ВМ для миграции.
3. Поиск целевого узла осуществляется не на всем наборе узлов (согласно описанию подхода).
4. Не говорится о том, как часто выполняется проверка необходимости миграции.

Алгоритмы не учитывают продолжительности нахождения узла в состоянии повышенной загруженности: очевидно, что при непродолжительной загруженности длительность миграции может снизить ее эффективность.

В настоящее время ведутся направления по разработке динамических подходов, основанных на работах [Никульчев, 2008; Lemmon, 2012]. Данные работы используют динамические модели в форме системы конечно-разностных моделей на основе идентифицированных моделей [Никульчев, 2004].

В схеме (рис. 3) исполнения и оптимизации динамических распределенных запросов в облачных пиринговых сетях авторами также разработан фреймворк (DObjects) для работы с p2p-сетями.

Ключевым элементом обработки запросов в предлагаемом подходе является наличие ядра, способного динамически адаптироваться к условиям сети и источникам. При этом подходе вывод результатов и физические расчеты по плану выполняются динамически и итеративно.

Такой подход гарантирует лучшую реакцию на изменение нагрузки и позволяет снизить задержки в системе. Следует отметить, что оптимизация исполнения запросов на локальных БД в текущем подходе ложится на адаптеры и источники данных.

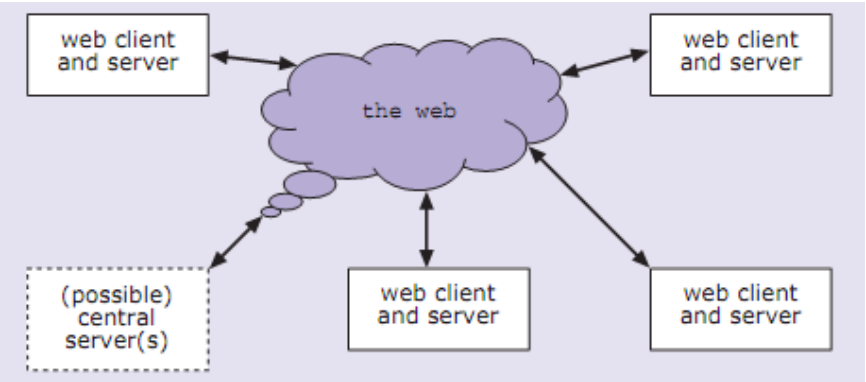

Рис. 3. Архитектура облачной пиринговой (p2p) сети

Исполнение и оптимизация запроса состоят из нескольких основных этапов.

1. В момент получения узлом запроса от пользователя генерируется высокоуровневый план исполнения.

2. На следующем шаге узел, исполняющий запрос, выбирает активные элементы плана сверху вниз в порядке следования. Однако исполнение активного элемента может быть делегировано любому узлу системы в целях достижения масштабирования нагрузки. Для выбора целевого узла исполнения в сети разворачивается модуль, способный адаптироваться к специфике сети и загруженности ресурсов. Если активный элемент передается на исполнение удаленному узлу, то управление его дочерними элементами также возлагается на этот узел. Удаленный узел, в свою очередь, может принять решение о перемещении дочерних узлов элемента плана на исполнение другим узлам либо выполнить локально.

Достоинством данного подхода является то, что он может быть внедрен в кратчайшие сроки, поскольку алгоритмы были реализованы в виде фреймворка. Однако облачные центры обработки данных, основанные на пиринговых сетях, на данный момент мало применяются на практике.

Заключение

Каждый из представленных методов обладает как достоинствами, так и недостатками. Общим для всех недостатком является синтетичность результатов, то есть то, что статистика по внедрению получена на искусственных системах, созданных только для тестирования подхода. Однако относительно большое количество исследований для довольно молодой области говорит о том, что проблема оптимизации актуальна и такие разработки в ближайшем времени будут востребованы. В настоящем сообщении рассматривается в широком смысле задача оптимизации обращений к базам данных и даются практические рекомендации, которые могут упростить обработку очень больших массивов данных.

Список литературы

Никульчев Е. В. Динамическое управление трафиком программно-конфигурируемых сетей в облачной инфраструктуре / Е. В. Никульчев, С. В. Паяин, Е. В. Плужник // Вестник Рязанского радиотехнического университета. — 2003. — № 3. — С. 54–57.

Никульчев Е. В. Использование групп симметрий для идентификации сложных систем / Е. В. Никульчев // Вычислительные технологии. — 2004. — Т. 9, № 3. — С. 72–80.

Никульчев Е. В. Построение модели загрузки каналов связи в сетях передачи данных на основе геометрического подхода / Е. В. Никульчев, С. В. Паяин // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. — 2008. — № 6. — С. 91–95.

Плужник Е. В. Слабоструктурированные базы данных в гибридной облачной инфраструктуре / Е. В. Плужник, Е. В. Никульчев // Современные проблемы науки и образования. 2013. — № 4. URL: www.science-education.ru/110-9980 (дата обращения: 15.10.2013).

Плужник Е. В. Функционирование образовательных систем в гибридной облачной инфраструктуре / Е. В. Плужник, Е. В. Никульчев // Известия вузов. Проблемы полиграфии и издательского дела. — 2013. — № 3. — С. 96–105.

Fegaras L. An Optimization Framework for Map-Reduce Queries / L. Fegaras, C. Li, U. Gupta // Proc. of the 15th International Conference on Extending Database Technology. — ACM, 2012. — P. 26–37.

Jahani E. Automatic optimization for MapReduce programs / E. Jahani, M. J. Cafarella, C. Ré // Proceedings of the VLDB Endowment. — 2011. — Vol. 4, No. 6. — P. 385–396.

Jurczyk P. Dynamic query processing for p2p data services in the cloud / P. Jurczyk, L. Xiong // Database and Expert Systems Applications. — Springer Berlin Heidelberg, 2009. — P. 396–411.

Lemmon M. D. Towards a passivity framework for power control and response time management in cloud computing // In Proc. of 7th Intl. Workshop on Feedback Computing, San Jose, CA. 2012.

Wang X. A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing / X. Wang, X. Liu, L. Fan, X. Jia // Mathematical Problems in Engineering [Электронный журнал]. — 2013. — Vol. 2013. — Режим доступа: http://www.hindawi.com/journals/mpe/2013/878542/, свободный. — Дата обращения 15.10.2013.

Zhang G. Massive Data Query Optimization on Large Clusters / G. Zhang, Chao LI, Yong Zhang, Chunxiao Xing. // Journal of Computational Information Systems. — 2012. — Vol. 8. — С. 3191–3198.