УДК 004.657

МЕТОДЫ ОПТИМИЗАЦИИ ПОИСКА В ОБЛАЧНЫХ БАЗАХ ДАННЫХ

Леонов Д. В.[1]

[1]*НОУ ВПО ОУ ВПО«Московский технологический институт«ВТУ», Москва, Россия(117292, г. Москва, ул. Кедрова, д. 8, кор. 2), e-mail: nir@mti.edu.ru*

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. Таким образом, внедрение облачных технологий определило появление новых подходов к оптимизации запросов в облачных базах данных. В статье проводится анализ эффективных методов по решению задачи оптимизации запросов. Выделены основные направления, позволяющие оптимизировать работу облачных баз данных — в облачных центрах обработки данных в формате SQL и новой парадигме построения запросов NoSQL. В статье рассмотрены архитектурно-зависимые решения, особенностью которых является использование динамического подхода. Приведены результаты экспериментального внедрения, рассмотрены достоинства и недостатки методов и подходов к оптимизации запросов.

Ключевые слова: облачные вычисления, облачные базы данных, методы оптимизации поиска, оптимизация запросов.

METHODS OF QUERY OPTIMIZATION IN CLOUD DATABASES

Leonov D.V.[1]

[1] *Moscow Technology Institute «VTU», Moscow, Russia (117292, Moscow, Kedrov St., 8, box. 2), e-mail: nir@mti.edu.ru*

Now growth of full range of services in the field of cloud computing is watched. This service is available for a large number of users. There is a whole new trend in the design of software applications focused on the processing of big data. Thus, the adoption of cloud technologies has defined the appearance of new approaches to query optimization in cloud databases. The article analyzes the effective methods to solve the query optimization tasks. The defines the main directions allowing optimizing queries to the cloud database: in cloud data centers in the SQL format and the new in the paradigm of building queries NoSQL. In the article the architecture-specific solutions, feature of which is the use of the dynamic approach. Results of experimental research are presented and the advantages and disadvantages of the approaches optimization query methods.

Keywords: cloud computing, cloud databases, methods of search optimization, query optimization

Введение

Существует множество определений термина «оптимизация запросов», наиболее часто встречающееся в литературе звучит следующим образом: под оптимизацией запросов понимается функция СУБД, осуществляющая поиск оптимального плана выполнения запросов из всех возможных для заданного запроса, либо процесс изменения запроса и/или структуры баз данных (БД) с целью уменьшения использования вычислительных ресурсов при выполнении запроса.

В настоящее время происходит развитие и распространение технологии «облачных вычислений» [1, 5]. 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 [4].

Целью данной работы является классификация и краткий анализ существующих разработок, которые могут применяться для оптимизации поиска в облачных БД.

Отметим, что топология сети также должна быть учтена при решении задачи оптимизации запросов в облачных системах хранения данных [1].

1. Оптимизация запросов в облачных БД SQL типа

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 [11] 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).

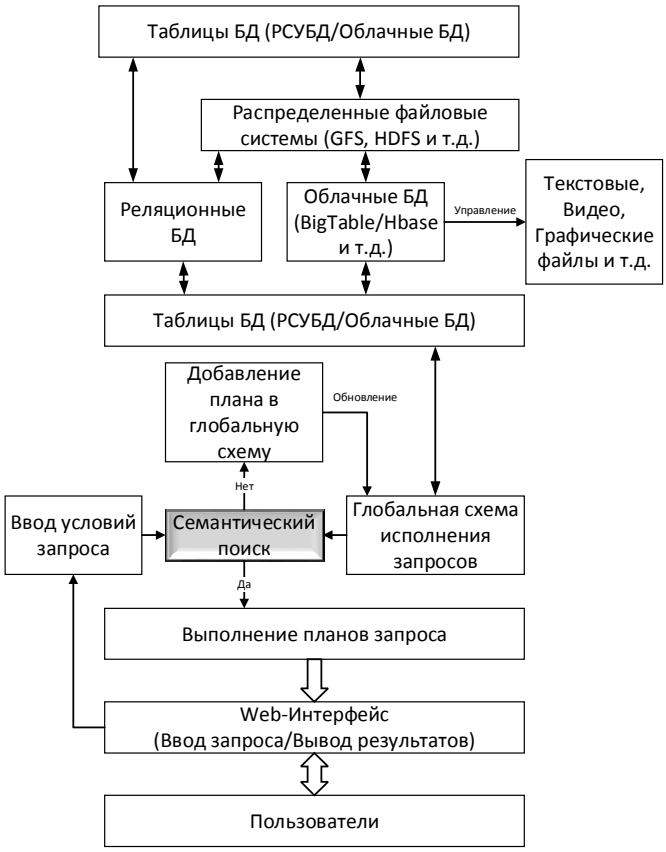

Рис. 1. Архитектура системы, основанной на SQL запросах

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).

Результаты экспериментального внедрения архитектуры, приведенные авторами в [11], показали четырехкратный прирост производительности, что говорит об эффективности используемых алгоритмов.

2. Оптимизация запросов в облачных базах данных NoSQL типа

Модель программирования map-reduce (MR) — популярная платформа для облачных вычислений, которая позволяет выполнять анализ больших объемов данных в облаке. MR облегчает параллельное выполнение специальных, длительных задач анализа больших объемов данных в кластере с shared-nothing архитектурой. Основная идея модели MR проста. Каждая задача MR представляется в виде map и reduce задания. Задание map указывает, каким образом будет обрабатываться пары ключ/значение для создания набора промежуточных пар, в то время как задание reduce определяет, как объединить все промежуточные значения, связанные одним промежуточным. Ядро MR для хранения и репликации данных использует распределенную файловую систему (DFS).

В основе подхода, предложенного в [6], лежит использование алгебры запросов и применение некоторых операторов высшего порядка, которые реализованы в существующих map-reduce системах (например, Hadoop). Отметим, что предлагаемый подход в первую очередь ориентирован на использование с языком MRQL. В отличие от других существующих map-reduce языков, таких как HiveQL и PigLatin, которые позволяют создавать скрипты с использованием недекларативных языков, MRQL достаточно выразителен и позволяет писать собственные скрипты для значительного круга задач в декларативной форме и в то же время поддается оптимизации.

Как и в случае с реляционными базами данных, целью оптимизации MRQL запросов является нахождение оптимального плана исполнения. Алгоритм оценки плана исполнения MRQL запроса состоит из следующих шагов:

1) Упрощение запроса.
2) Построение графа запроса.
3) Представление графа запроса в алгебраической форме.

4) Формирование карты алгебраической формы для оценки и улучшения плана с использованием методом алгебраической оптимизации.

5) Создание функции MR сочетания на основе MR функции *reduce*.

Достоинством данного подхода является то, что разработанные алгоритмы реализованы в виде фреймворка, исходный код которого находится в свободном доступе. При этом проект на данный момент активно развивается.

Система Manimal – еще одна разработка, нацеленная на оптимизацию map-reduce программ [7]. В основе системы лежит механизм статического анализа кода для обнаружения блоков, которые могут быть оптимизированы. По заявлению авторов, как и большинство оптимизаторов языков программирования, Manimal является максимально-эффективной системой, но, тем не менее, это не гарантирует, что будут обнаружены все существующие блоки, которые могут быть оптимизированы, поскольку всегда есть возможность вести разработку таким образом, что результат в дальнейшем невозможно будет оптимизировать автоматически.

Manimal состоит из трех основных компонентов, которые позволяют полностью автоматизировать процесс оптимизации map-reduce программ. Анализатор рассматривает представленные пользователем map-reduce программы и отправляет оптимизатору полученный в результате дескриптор оптимизации. Оптимизатор использует дескриптор оптимизации с предварительно вычисляемыми параметрами, хранящимися в определенном каталоге, для того, чтобы выбрать оптимальный план исполнения, результатом является дескриптор исполнения. Этот дескриптор вместе с копией оригинальной программы отправляется фабрике исполнения. Отметим, что фабрика исполнения сохраняет стандартную map-shuffle-reduce последовательность.

Согласно [7] Manimal для оптимизации использует три различных подхода:

1) Оптимизация выборки (selection). Выборки в коде являются результатом выполнения функции map(), которая отрабатывает только при определенных условиях, налагаемых на связанные параметры. Как и в случае с реляционной выборкой, не имеет смысла обрабатывать данные, не удовлетворяющие налагаемым условиям, любой вызов функции map() для таких данных будет бессмысленным. Как в случае реляционной выборки, оптимизация может быть выполнена с помощью B-деревьев. Существенным достижением *Manimal* является то, что система может автоматически определять выборки, удовлетворяющие условиям, описанным выше.

2) Оптимизация проекции (projection). Оптимизация проекций изменяет файлы данных на диске таким образом, чтобы хранились только байты, которые действительно необходимы

для выполнения пользовательского кода. Сжатие исходного файла путем удаления неиспользуемых в работе полей сократит объем файла и позволит его обработать.

3) Оптимизация сжатия данных. Сжатие данных отличается от поддерживаемого в Hadoop сжатия. Hadoop хранит на диске сжатые версии входных данных и их промежуточные образы. Распаковка выполняется непосредственно перед функциями map() и reduce(). При этом Hadoop использует единую методику сжатия данных для всех файлов, в то время как *Manimal* позволяет использовать две семантически связанные формы сжатия: дельта-сжатие и работу непосредственно со сжатыми данными.

Результаты экспериментального внедрения Manimal на кластере из пяти узлов в двух случаях из четырех показали производительность, соизмеримую с производительностью реляционных СУБД [7]. При этом большинство обнаруженных возможностей для оптимизации не дали существенного выигрыша в производительности. На основании экспериментов можно сделать вывод о том, что система Manimal обладает потенциалом, однако требует дальнейшего развития.

3. Архитектурно-зависимые решения

Децентрализованное управление миграцией виртуальных машин в крупномасштабных центрах облачной обработки предложено в [10]. 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 (очевидно, что скорость, с которой «облако» отвечает на запросы пользователя, является одним из критериев 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. В [10] приводится описание децентрализованного механизма, позволяющего избежать указанной проблемы.

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 (в отличие от схемы с централизованным управлением, где все пакеты должны проходить через общий узел).

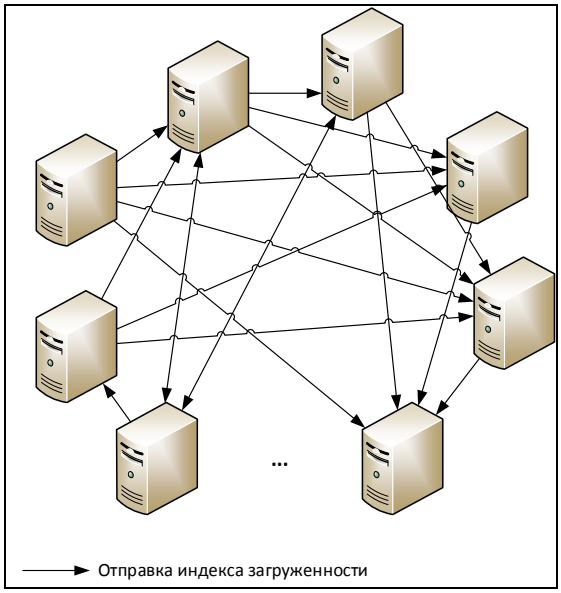

Рис. 2. Децентрализованный обмен индексом загруженности.

Индекс загруженности узла представляет собой кортеж следующего вида:

$$LI = <src, dest, util> ,$$

где *src* – идентификатор узла, откуда был получен индекс, *dest* – содержит идентификатор узла, который получит индекс, *util* – использование процессора узлом-источником (*src*).

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 [10]. 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.

Следует отметить, что недостатками рассматриваемого подхода является:

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 (по крайней мере, в том виде, в котором они приведены в [11]) 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 [3, 9]. Данные работы используют динамические модели в форме системы конечно-разностных моделей на основе идентифицированных моделей [2].

В работе [8] рассматривается схема исполнения и оптимизации динамических распределенных запросов в облачных пиринговых сетях (рис. 3), 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.

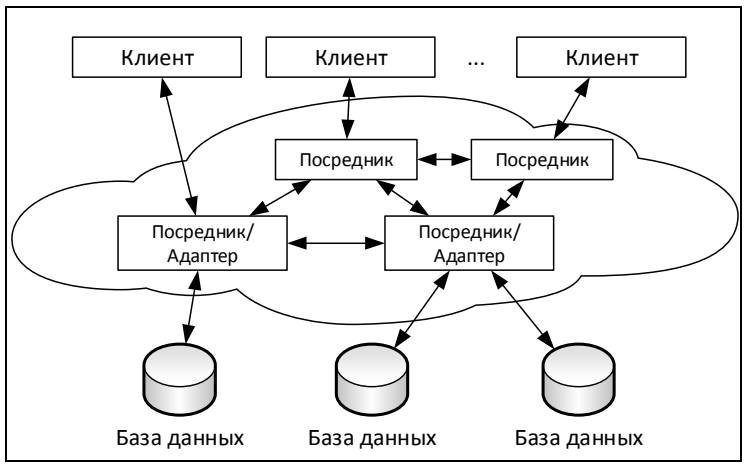

Рис. 3. Архитектура облачной пиринговой (p2p) сети.

Исполнение и оптимизация запроса состоит из нескольких основных этапов:

1. В момент получение 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 так же 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 (набора библиотек). 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.

Список литературы

1. 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.
2. Никульчев Е.В. Использование групп симметрий для идентификации сложных систем / Е.В. Никульчев // Вычислительные технологии.— 2004.— Т. 9. - № 3.— С. 72-80.
3. 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.
4. 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.
5. 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.
6. 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.
7. Jahani E. Automatic optimization for MapReduce programs / E. Jahani, M. J. Cafarella, C. Ré // Proceedings of the VLDB Endowment. – 2011. – V. 4. - N. 6. – P. 385-396.
8. 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.
9. 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.
10. 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. – V. 2013. — Режим доступа: http://www.hindawi.com/journals/mpe/2013/878542/ , свободный. — Дата обращения 15.10.2013.
11. Zhang G. Massive Data Query Optimization on Large Clusters / G. Zhang, Chao LI, Yong Zhang, Chunxiao Xing. // Journal of Computational Information Systems. — 2012. – V. 8. – C. 3191–3198.

Рецензенты:

Винокур А.И., д.т.н., профессор, директор института принтмедиа и информационных технология ФГБОУ ПВО Московский государственный университет печати имени Ивана Федорова, г. Москва.

Никульчев Е.В., д.т.н., профессор, проректор по научной работе НОУ ВПО Московский технологический институт «ВТУ», г. Москва.