А.В. СМИРНОВ

Распределенный подход к поиску в пространстве оптимизатора запросов

УДК 004.657

Муромский институт (филиал) ФГБОУ ВПО «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых», г. Муром

В статье рассматривается распределенный алгоритм оптимизации запроса в кластере.

The article describes distributed algorithm of query optimization in cluster.

Современные задачи по обработке данных — тяжелое испытание существующих СУБД. Проблема обработки сверхбольших объемов данных, требует новых решений, готовых к петабайтной масштабируемости.

В астрономии, физике высоких энергий, в науках о земле вопрос о подходящей системе хранения и обработке научных данных встает особенно остро.

Например в российском космическом проекте «ЛИРА» за 2 года будет принято 200-300 ТБ данных с ПЗС матриц [1]. А американский астрономический проект LSST произведет порядка 6 ПБ в год [2].

Самый известный проект в области физики высоких энергий, Большой Адронный Коллайдер, произведет 4 Пт данных в год.

Разработка СУБД затрагивает большое количество связанных между собой областей исследования, таких как обработка запросов, хранение данных, выполнение, сетевое взаимодействие.

Одним из важных компонентов запросов является оптимизатор запросов.

В условиях обработки сверхбольшого набора данных качество оптимизации запроса встает особенно остро, так как в конечном счете производительность, а соответственно и время выполнения зависит от того, какой план выбран.

Решая задачу оптимизации приходится находить компромисс между двумя взаимосвязанными проблемами — скоростью оптимизации и качеством оптимизации. Так как задача оптимизации запросов математически не формализуема, основной подход в ее решении — перебор множества вариантов выполнения запросов. Очевидно, что в общем случае все варианты перебрать невозможно, так как количество их слишком велико и растет экспоненциально сложности запроса, поэтому приходится использовать методы усечения количества вариантов, чтобы сократить выбор. Таким образом допускается, что получившийся, более производительный результат приемлем и считается оптимальным.

Рассматривая существующие работы по оптимизации запросов в распределенных СУБД можно отметить, что данная проблема решается в два этапа:

1. Статическая оптимизация запроса — традиционная оптимизация на узле-координаторе кластера, не учитывающая динамическую составляющую работы РСУБД (например уровень загрузки сети или узлов кластера);

2. Динамическая оптимизация запроса — оптимизация во время выполнения запроса, основа которой оптимальное распределение веток запроса в кластере на основе динамической составляющей [3].

Проблема в том, что от плана выполнения, полученного на этапе статической оптимизации, в значительной степени зависит успешность дальнейшего выполнения запроса. Так, например выбранный алгоритм доступа соединения таблиц уже не может быть изменен в процессе выполнения и худший вариант не может быть исправлен балансировкой запроса.

В данной работе предлагается использовать мощности кластера не только для выполнения запроса, но и для выполнения статической оптимизации. Используя распределенный механизм оптимизации запросов анализ вариантов выполнения может быть улучшен в сторону качества оптимизации запросов, но в то же время без увеличения времени оптимизации.

Основой предложенного подхода является целеориентированность оптимизатора запроса, использование распределенной мемо-

изации и кэширование. Рассмотрим каждую из этих составляющих по-отдельности, а затем в контексте модели оптимизатора.

Подобная техника используется во многих оптимизаторов запросов в особенности в оптимизационных фреймворках, так как позволяет обеспечить гибкость и расширяемость оптимизатора запросов. Подробно с ними можно ознакомиться в работах [4, 5]. В нашем случае это основа работы распределенного механизма.

Основой целеориентированности в оптимизаторе является очередь целей или заданий. Самое первое цель является заданием оптимизации группы верхнего уровня и все последующие задания генерируются отталкиваясь от него. Для реализации распределенной оптимизации были использованы несколько независимых очередей заданий, по одной на каждый узел кластера.

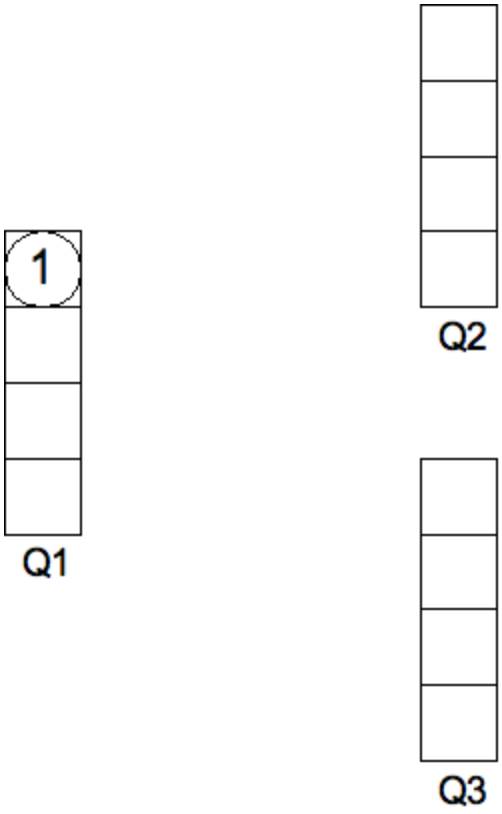

Рис. 1. Начальное состояние задачи оптимизации

Первое задание генерируется на узле-координаторе, куда поступает запрос пользователя, в дальнейшем распределение работы организовывается по принципам P2P [6], то есть каждый узел может передать работу работу любому доступному. На рисунке 1 показано начальное состояние очередей. Предположим, что мы имеем три

узла. Каждое задание помечается в очереди кругом. Стрелки между ними — направление перемещение задания. Во время выполнения цели №1 генерируются новые цели №2, №3 и №4, одна из которых добавляется в локальную очередь, а две других распределяются между свободными узлами (рис. 2)

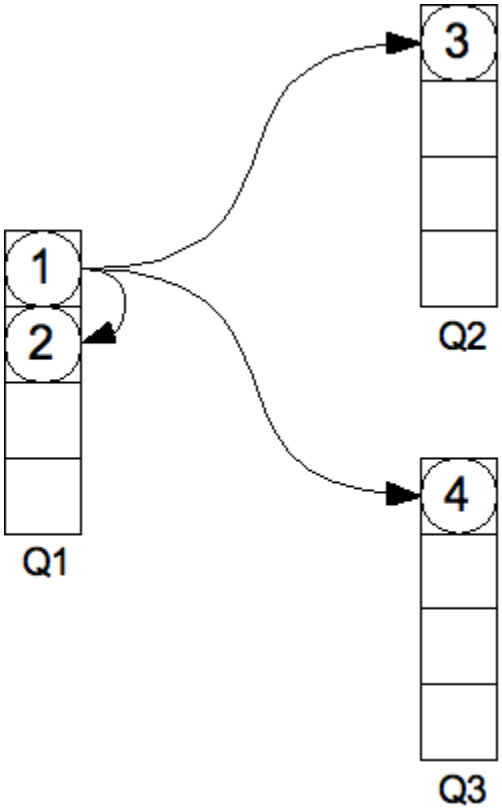

Рис. 2. Генерация и распределение целей

Таким образом каждое новое оконченное задание добавляет в кластер новые задания. Распределение заданий планируется с учетом активных количества заданий в кластере, а также учитывая «перспективность задания» [5], ставя наиболее перспективные в локальную очередь и передавая менее перспективные на удаленные узлы. Это позволяет равномерно распределить выполнение оптимизации между всеми узлами кластера.

После каждого выполненного задания, кроме добавления новых заданий в локальную и удаленные очереди выполняется перепланирование, которое заключается в 2х моментах:

1. Рекластеризация задачи оптимизации — перемещение заданий между узлами, для обеспечения более сбалансированного выполнения целей, так как некоторые задания могут выполняться

быстрее других, тем самым внося перекос в количестве заданий на разных узлах;

2. Сортировка заданий в очереди — перемещение заданий внутри очередей, для достижения выполнения некоторых целей, так например одно и то же задание может выполняться на разных узлах, с целью оптимизации мемоизация не позволяет выполнять одно и то же задание дважды, поэтому узел будет дожидаться тот, который поставил это задание первым, такое задание может быть перемещено выше в очереди, например если оно перспективное (рис. 3).

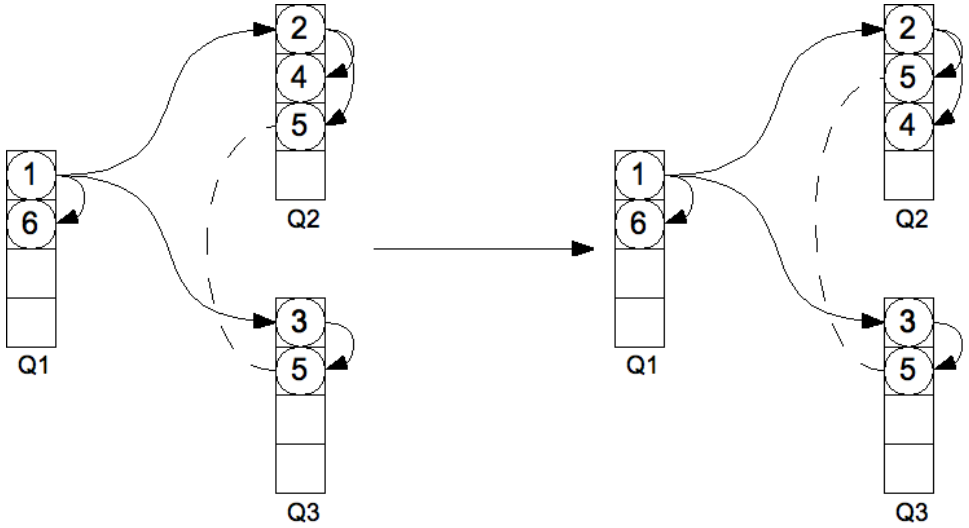

Рис. 3. Перепланирование заданий в очереди

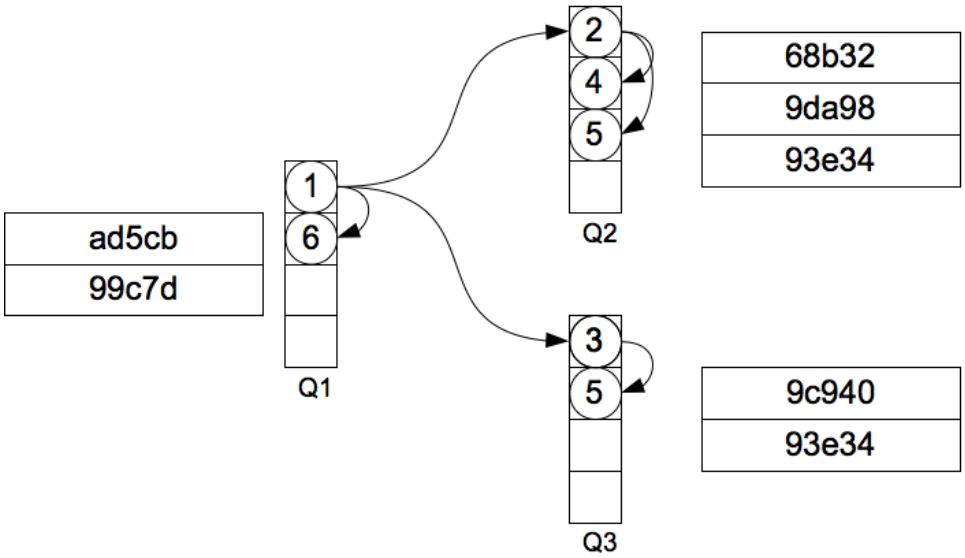

Рис. 4. Мемоизация на основе распределенной хеш-таблицы

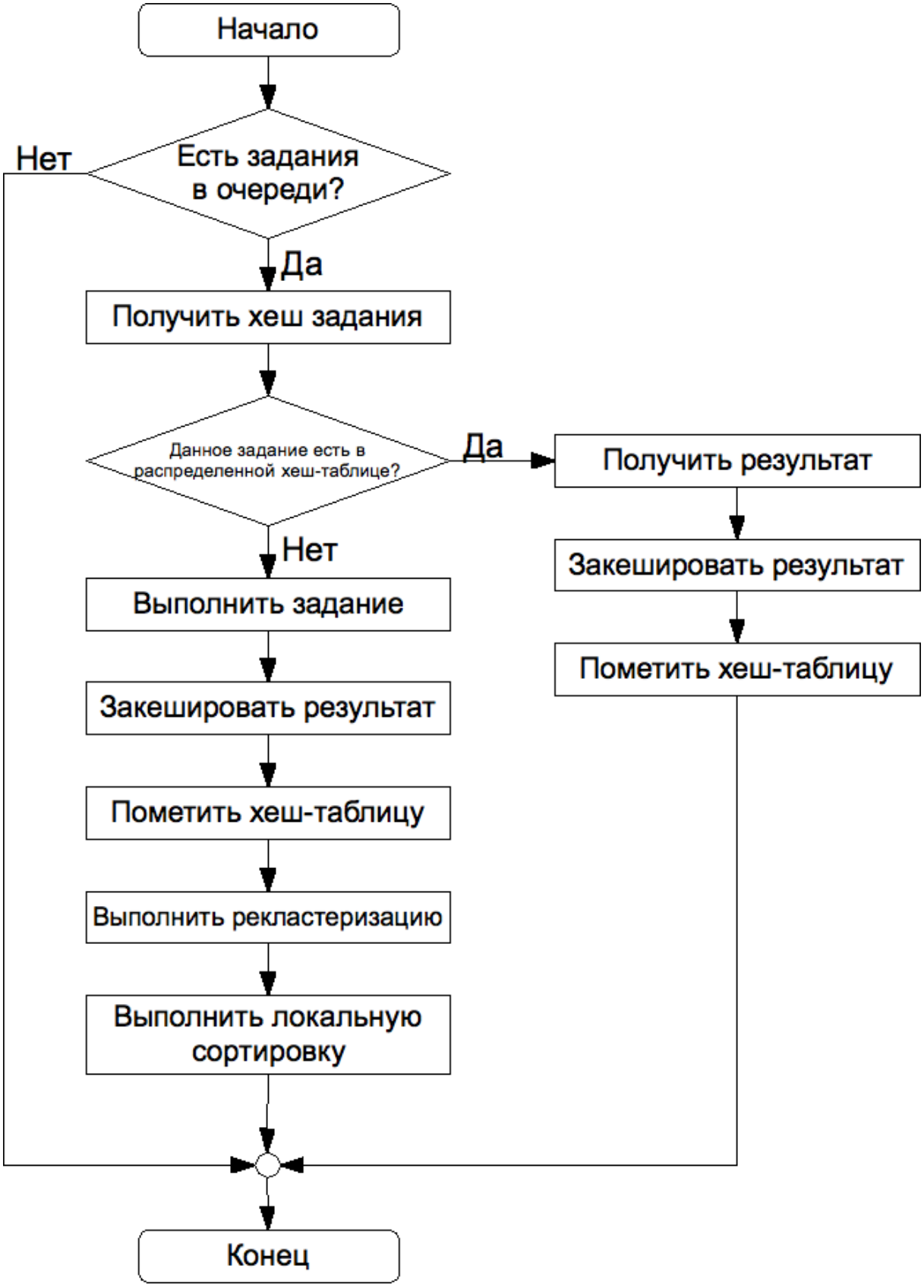

Рис. 5. Общий алгоритм работы

Для обеспечения эффективного выполнения заданий было предусмотрено предотвращение повторного выполнения заданий.

Для этого каждое задание перед выполнением получает хеш, основанный на типе выполняемого задания и на подзапросе, которое является входными данными для него, результат данного задания также маркируется данным хешем.

Каждый узел выполняя задание кэширует результат, который получил до этого, таким образом получая таблицу результат-хеш (рис. 4). Данные таблицы составляют общую распределенную хеш-таблицу, благодаря которой можно получить мемоизированный результат любого узла.

Узел, который запрашивает мемоизированный результат также кеширует его у себя, для использования в будущем, и помечая его в локальной хеш-таблице информирует другие узлы о наличие этого результата у себя. Однако кеширование результата с удаленных узлов носит временный характер и может со временем вытесняться, чтобы не вызывать перекоса хранимых данных в кластере.

На рисунке 5 показан полностью алгоритм работы каждого из узла кластера. Отметим, что только узел-координатор начинает работу с заданием в очереди, которое добавляет непосредственно оптимизатор (так как нет других заданий, которые могли бы поставить это задание в очередь), остальные же узлы имеют пустую очередь и получают задания в процессе генерации или рекластеризации.

Рассмотренный подход к организации оптимизации запросов в распределенных СУБД обладает следующим рядом качеств:

1. Крайне прост по структуре и использует известные методы организации распределенных вычислений;
2. Работает с существующими алгоритмами поиска в пространстве оптимизатора основанными на целеориентированных алгоритмах;
3. Интегрируется с существующей моделью статической-динамической оптимизации в распределенных СУБД, так как сам является расширением статической оптимизации;
4. Скорость оптимизации увеличивается линейно с ростом количества задействованных узлов, при неизменном способе усечения пространства поиска оптимизатора. Очевидно, что этого недостаточно для борьбы с экспоненциально увеличивающимся про-

странством поиска, при увеличении сложности запроса, тем не менее это дает положительный эффект;

5. Расширение пространства поиска при прямопропорциональном увеличении количества задействованных узлов не влияет на время оптимизации.

Литература

1. В ГАИШ МГУ разрабатывают новый космический телескоп для установки на МКС // MsuNews URL: http://www.msunews.ru/news/2683/.

2. LSST Science FAQ's. // URL: http://www.lsst.org/lsst/faq-science.

3. High Performance Parallel Database Processing and Grid Databases / *D. Taniar, C. H. C. Leung, W. Rahayu, S. Goel*. — Wiley Publishing, 2008. — Pp. 256–291.

4. *Kabra, N.* Opt++ : an object-oriented implementation for extensible database query optimization / N. Kabra, D. J. DeWitt // The VLDB Journal. — 1999. — April. — Vol. 8. — Pp. 55–78. http://dx.doi.org/10. 1007/s007780050074.

5. *Graefe, G.* The cascades framework for query optimization / G. Graefe // IEEE Data Eng. Bull. — 1995. — Vol. 18, no. 3. — Pp. 19–29.

6. Glossary of peer-to-peer terminology. // URL: http://www.p2pna.com/glossary. html.

E-MAIL: SMIRNOFFJR@GMAIL.COM

ТЕЛЕФОН: +7920944330

НАУЧНЫЙ РУКОВОДИТЕЛЬ:

Д.Т.Н. АНДРИАНОВ Д.Е.