в B&S-дерево, которое обладает той же шириной и $O(n)$ узлами. Теоретически обоснованы следующие достоинства B&S-дерева: переход от бинарного корневого дерева декомпозиции к B&S-дереву увеличивает число таблиц не более чем в два раза; B&S-дерево позволяет удерживать размер всякой таблицы динамического программирования в пределах оценки $O(kq^k)$, где q — число состояний, в которых может находиться вершина графа по отношению к возможному решению. Кроме того, для вычисления таблиц динамического программирования по B&S-дереву возможно привлечение аппарата реляционной алгебры и свойств ациклических баз данных. Это способствует алгоритмической конкретизации метода и сокращению числа строк в промежуточных таблицах (за счет использования монотонного плана соединения и программы полусоединений таблиц). Приведена демонстрация применения метода динамического программирования по B&S-дереву при решении оптимизационных задач (на примере задачи о вершинном покрытии) и задач удовлетворения ограничений (на примере задачи SAT).

Подробное изложение представленных результатов можно найти в [7].

ЛИТЕРАТУРА

1. *Downey R. and Fellows M.* Parameterized complexity. New York: Springer Verlag, 1999.
2. *Быкова В. В.* FPT-алгоритмы и их классификация на основе эластичности // Прикладная дискретная математика. 2011. № 2(12). С. 40–48.
3. *Bodlaender H. L.* Treewidth: Algorithmic techniques and results // LNCS. 1997. V. 1295. P. 19–36.
4. *Niedermeier R. and Rossmanith P.* A general method to speed up fixed-parameter-tractable algorithms // Inform. Proc. Lett. 2000. V. 73. P. 125–129.
5. *Быкова В. В.* Вычислительные аспекты древовидной ширины графа // Прикладная дискретная математика. 2011. № 3(13). С. 65–79.
6. *Courcelle B.* The monadic second-order logic of graphs. III. Tree decompositions, minors and complexity issues // RAIRO Inform. Theor. Appl. 1992. No. 26(3). P. 257–286.
7. *Быкова В. В.* FPT-алгоритмы на графах ограниченной древовидной ширины // Прикладная дискретная математика. 2012. № 2(16). С. 65–78.

УДК 004.65

ЭЛЕМЕНТЫ РАСПРЕДЕЛЁННОЙ СУБД НА БАЗЕ СЕРВЕРА MariaDB[1]

И. Н. Глотов, С. В. Овсянников, В. Н. Тренькаев

В связи с интенсивным развитием и появлением новых интернет-технологий, таких, например, как «облачные» вычисления, остаётся актуальной задача распределённой обработки больших объёмов данных в режиме реального времени. Так, имеется необходимость в распределённой СУБД, т. е. системе управления, распределённой в компьютерной сети базы данных (БД) [1]. При этом распределённая БД должна обеспечивать прозрачный доступ к данным, т. е. должна выглядеть с точки зрения пользователей и прикладных программ как централизованная БД.

Наиболее востребованными свойствами распределённых СУБД, ориентированных на использование в «облачных» сервисах, для которых характерна работа с большими

[1]Работа выполнена в рамках реализации соглашения о сотрудничестве между кафедрой защиты информации и криптографии ТГУ и ООО «Хайвекс Технолоджи» по проекту JelasticDB.

объёмами данных, большим количеством пользователей, а также работа в условиях резко изменяющейся нагрузки (возрастание объёма данных и/или потребностей их обработки), считается масштабирование по требованию и отказоустойчивость. Под масштабированием понимается способность системы при увеличении рабочих нагрузок увеличивать свою производительность за счёт увеличения используемых ресурсов.

В настоящее время существует множество распределённых СУБД со свойством масштабируемости [2], большую часть которых можно отнести к так называемому NoSQL-подходу, в рамках которого не используются реляционные схемы данных. Данное исследование направлено на разработку принципов построения системы управления реляционной MySQL-совместимой распределённой БД со свойствами масштабируемости и отказоустойчивости.

За основу взята открытая СУБД MariaDB (ветка MySQL), которая изначально ориентирована на управление локальными данными. Использование СУБД с открытым кодом позволяет избежать больших трудозатрат при построении экспериментального стенда по изучению характеристик разрабатываемой СУБД (время отклика на запрос, процент использования ресурсов и др.). Кроме того, сервер MariaDB (MySQL) отличается высокой производительностью и является довольно популярным при использовании как в коммерческих решениях, так и в учебных целях.

Рассматривается разновидность многозвенной архитектуры клиент-сервер, когда функция обработки (хранения) данных вынесена на несколько отдельных серверов. Клиентские приложения обращаются не напрямую к серверам БД, но через промежуточный слой. СУБД располагается в компьютерной сети с тремя уровнями узлов: балансировки нагрузки (балансировщики); приёма и обработки SQL-запросов (SQL-узлы); обеспечения физического доступа к данным (узлы хранилища). Запросы от клиентов направляются через балансировщика нагрузки на одну из доступных и наименее загруженных машин сети для обработки SQL-запроса. SQL-узел, в свою очередь, обращается к узлу хранилища, на котором находятся требуемые данные. Балансировщик выявляет также отказавшие сервера и перенаправляет трафик SQL-запросов на рабочие узлы. Для повышения надёжности и увеличения быстродействия данные тиражируются по нескольким узлам хранилища с использованием механизма репликации. Для обеспечения обработки данных большого объёма используется механизм фрагментации, т. е. разбиение на части и распределение по отдельным узлам хранилища. В качестве узлов балансировки нагрузки используются MySQL proxy, в качестве SQL-узлов и узлов хранилища — серверы MariaDB.

Такая архитектура обеспечивает горизонтальную масштабируемость и эластичность на всех уровнях, поскольку возможно увеличение (уменьшение) числа узлов любого типа для повышения производительности системы при возрастании (уменьшении) рабочей нагрузки или для решения проблем, связанных с изменением размеров БД.

В традиционном варианте сервер MariaDB включает в себя четыре основных модуля [3]: кэш SQL-запросов; синтаксический анализатор SQL-запросов; оптимизатор SQL-запросов; движок хранилища данных (*storage engine*, далее — движок). Совокупность модулей сервера MariaDB, за исключением движка, будем называть ядром. Обращение к движку происходит посредством низкоуровневого storage API, представляющего собой список функций, выполняющих простые операции с таблицей. Движок преобразует запросы к нему в операции чтения-записи диска. Кроме того, движок может поддерживать много дополнительных возможностей, таких, как создание и поддержка структуры индексов, обеспечение ACID транзакций и др. СУБД MariaDB

позволяет в зависимости от конкретных нужд приложения использовать различные движки хранилища данных, которых на данный момент существует более десятка [4].

Особенностью предлагаемого решения является использование удалённого движка, который располагается на узле хранилища; в качестве него можно взять любой MySQL-совместимый движок. В типовом случае сервер MariaDB устанавливается на одной машине, на которой располагается и система хранения. В нашем случае сервер MariaDB разделен на две части: ядро и движок. При этом движок вынесен на удалённый узел, который по сети принимает запросы от ядра. Для реализации данной схемы разработаны дополнительные модули для сервера MariaDB: виртуальный движок и виртуальное ядро, а также протокол VSVD сетевого взаимодействия этих модулей.

На SQL-узле устанавливается сервер MariaDB, использующий виртуальный движок, а на узле хранилища — сервер MariaDB с загруженным реальным движком, но виртуальным ядром. SQL-узел не хранит данных, кроме описания БД и таблиц. Виртуальный движок и виртуальное ядро нужны для состыковки реальных ядра и удалённого движка, чтобы ядро сервера MariaDB SQL-узла работало с удалённым движком сервера MariaDB узла хранилища и наоборот, не замечая, что на самом деле они взаимодействуют через сеть. Виртуальный движок перенаправляет запросы на узел хранилища, где виртуальное ядро принимает эти запросы и обращается с ними к реальному движку. Более подробно: каждый SQL-запрос в ядре MariaDB превращается в серию вызовов функций storage API, которые группируются в одну функцию VSVD API, сериализуются и передаются на соответствующий узел хранилища, где формируется ответ и таким же образом отправляется назад. Тем самым протокол VSVD схож с протоколом RPC по характеру своей работы (удалённый вызов процедуры). На сервере MariaDB узла хранилища стандартный MySQL-протокол отключен, используется протокол VSVD для приема/передачи запросов VSVD API.

Необходимой составляющей любого SQL-узла является базовый модуль, отвечающий за репликацию и фрагментацию данных, управление одновременным доступом к данным, маршрутизацию storage API запросов, масштабирование и отказоустойчивость на уровне узлов хранилища.

На данный момент реализована сетевая подсистема распределённой СУБД на основе разделения сервера MariaDB на ядро и удалённый движок. Результаты работы могут быть использованы для построения экспериментального образца реляционной масштабируемой распределённой СУБД, для чего необходимо разработать соответствующее алгоритмическое и программное обеспечение для базового модуля SQL-узла.

ЛИТЕРАТУРА

1. *Коннолли Т., Бегг К.* Базы данных. Проектирование, реализация и сопровождение. Теория и практика. М.: Вильямс, 2003. 1440 с.
2. *Cattell R.* Scalable SQL and NoSQL data stores // ACM SIGMOD Record. 2010. V. 39. Iss. 04. P. 12–28.
3. *Pachev S.* Understanding MySQL Internals. O'Reilly Media, 2007. 234 p.
4. MySQL Storage Engine Comparison Guide. June 2009. `http://www.mysql.com/why-mysql/white-papers`.