УДК 004.657 DOI 10.25559/SITITO.2017.3.521

Мунерман В.И., Мунерман Д.В.

Смоленский государственный университет, г. Смоленск, Россия

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ СИММЕТРИЧНОГО ГОРИЗОНТАЛЬНОГО РАСПРЕДЕЛЕНИЯ ДАННЫХ НА ОСНОВЕ СЕТЕВЫХ ТЕХНОЛОГИЙ

Аннотация

В работе рассмотрен подход к построению программно-аппаратных комплексов для реализации симметричного горизонтального распределения таблиц в реляционных базах данных. Для решения поставленной задачи предложено использовать стандартные технологии разработки сетевого программного обеспечения. Показано, что наиболее эффективным методом служит построение (WEB API)-приложения. Это позволяет независимо от платформы, использовать средства разработки, подобные ASP.NET Framework и паттерны технологии программирования MVC. Дано описание методов построения такого программного обеспечения. На основе этого подхода сделана сетевая реализация алгоритма бустрофедона, с помощью которого выполняется симметричное горизонтальное распределение таблиц в реляционной базе данных. Приведены результаты вычислительного эксперимента, показавшего эффективность предложенного подхода.

Ключевые слова

Массовая обработка данных; параллельное программирование; программно-аппаратные комплексы; WebAPI.

Munerman V.I., Munerman D.V.

Smolensk State University, Smolensk, Russia

PARALLEL IMPLEMENTATION OF SYMMETRIC HORIZONTAL DISTRIBUTION OF DATA ON THE NETWORK TECHNOLOGIES BASIS

Abstract

The approach to the construction of software and hardware complexes for implementing a symmetric horizontal distribution of tables in relational databases is considered. To solve this problem, we propose to use standard technologies for the development of network-software. It is shown that the most effective method is the construction (WEB API)-application. This allows regardless of the platform, to use development tools similar to the ASP.NET Framework and the patterns of MVC programming technology. A description of the methods of building such software is given. On the basis of this approach, a network implementation of the Boustrophedon algorithm is made, with the help of which a symmetrical horizontal distribution of tables in a relational database is performed. The results of a computational experiment showing the effectiveness of the proposed approach are presented.

Keywords

Massively data processing; parallel programming; software-hardware complex; WebAPI.

Для параллельной реализации операции Join, которая в теоретико-множественной (файловой) модели данных определяется как слияние нестрого упорядоченных файлов, необходимо специальным образом распределить таблицы между процессорами, выполняющими операцию над фрагментами таблиц. Для этой цели используется принцип симметричного горизонтального распределения данных, подробно рассмотренный в [1]. Для реализации этого принципа исходные данные – таблицы-операнды должны быть представлены как индексно-последовательные. Это означает, что каждой таблице соответствует ***индекс***, каждая

запись которого содержит: значение ключа, задающего сравнение строк таблиц-операндов при выполнении Join; номер первой строки с таким значением ключа; количество строк с этим значением ключа. Из ***индексов*** обеих таблиц строится ***индекс распределения*** как пересечение ***индексов*** таблиц-операндов, каждая строка которого содержат данные соответствующих строк обоих индексов. На основе ***индекса распределения*** производится распределение данных таблиц-операндов между фрагмент-процессорами, выполняющими операцию Join над фрагментами таблиц операндов. При этом, для достижения максимального эффекта от распараллеливания, необходимо, чтобы количества записей во всех фрагментах таблиц были наиболее близки друг другу. Для достижения этой цели используется один из оптимизационных алгоритмов [1, 2], например, мультипликативный рюкзак (Multiple Knapsack) или алгоритм бустрофедона.

Для параллельной реализации операции Join может быть использован программно-аппаратный комплекс на основе вычислительной сети: локальная сеть, интранет, интернет. Одно из решений для создания такого комплекса состоит в следующем:

1. В вычислительной сети выделяется один узел, которому назначается роль хост-машины. Этот узел связан с основной базой данных (БД), в которой содержатся таблицы-операнды операции Join;
2. Выделяется некоторое количество узлов сети, которые связанны с фрагментами основной БД (БД-фрагментами). Этим узлам назначается роль серверов баз данных, в дальнейшем они будут называться S-серверами;
3. Программное обеспечение хост-машины представляет собой совокупность приложений, которые реализуют взаимодействие с основной БД, формируют индексные таблицы для таблиц операндов и их фрагментов, осуществляют взаимодействие с S-серверами;
4. Программное обеспечение S-сервера представляет собой web-сервис, реализующий две основные функции: прием от хост-машины фрагментов таблиц и формирование таблиц в БД-фрагменте основной БД и выполнение операции Join в этом БД-фрагменте с возвратом ее результата хост-машине.

Схема передачи данных между вычислительными узлами такого комплекса показана на рисунке 1.

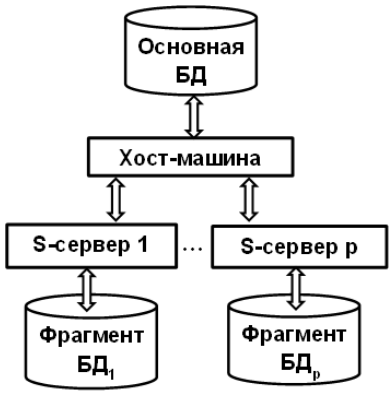

Рис. 1. Схема передачи данных между хост-машиной и S-серверами

В рассматриваемом случае важную роль играет выбор архитектуры и технологии разработки программного обеспечения, задача которого состоит в передаче данных между основной БД и ее фрагментами. Одно из решений состоит в выборе архитектуры (WEB API)-приложения, которая позволяет, как правило, независимо от платформы, использовать средства разработки, подобные ASP.NET Framework и паттерны технологии программирования MVC [3]. Некоторая особенность применения такой архитектуры для разработки предлагаемого программно-аппаратного комплекса состоит в отличии распределения ролей узлов сети от традиционно принятого в клиент-серверной архитектуре. А именно, хост-машина реализует функцию единственного клиента, который обеспечивает связь с основной БД и взаимодействует по данным и управлению с S-серверами, реализующими обработку фрагментов WEB-сервисами (в дальнейшем S-сервисами). То есть, предложенная архитектура программно-аппаратного комплекса представляет собой сеть с единственным клиентом и множеством серверов, которые выполняют единый набор функций. Такое разделение функций между вычислительными узлами сети имеет логический характер и определяется типом программного обеспечения, которое сопоставлено узлу. Это может быть программное обеспечение хост-машины или S-сервис.

На рисунке 2 показано, какие функции и каким образом выполняет программное обеспечение хост- машины (клиента) при реализации симметричного горизонтального распределения данных.

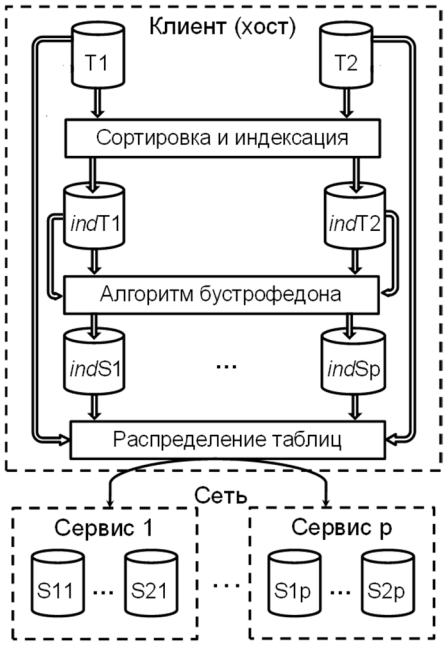

Рис. 2. Функциональная схема программного обеспечения хост-машины

Процедура сортировки и индексации считывает построчно проекции таблиц-операндов (на рисунке T1 и T2), содержащие только ключ, по которому реализуется операция Join. Таблицы T1 и T2 упорядочены по этому ключу. В процессе считывания, для каждого значения ключа формируется строка индексной таблицы, содержащая это значение, соответствующее ему количество строк таблицы-операнда и, при необходимости, номер первой строки таблицы-операнда с этим значением ключа. Если использовать только те средства, которые реализованы в БД, то эти результаты могут быть получены запросами вида

SELECT <Key>, SUM(1) AS M FROM T_i GROUP BY <Key> ORDER BY <Key>, (i=1, 2).

Для этих запросов в БД создаются хранимые процедуры. Поскольку считывание из БД и выполнение запросов современные СУБД могут производить параллельно, то эту функцию целесообразно запускать в двух параллельных потоках. Результат работы этой процедуры две индексных таблицы *ind* T1 и *ind* T2.

Процедура, реализующая алгоритма бустрофедона, начинается формированием сводной индексной таблицы, строки которой содержат значение ключа и произведение количества строк с этим значением ключа из индексных таблиц *ind* T1 и *ind* T2. Формирование сводной таблицы может быть реализовано запросом

SELECT *ind* T1.<Key> AS <Key>, *ind* T1.M* *ind* T2.M AS MM
FROM ind T1 INNER JOIN ind T2 ON ind T1.<Key>= ind T2.<Key>
ORDER BY <Key>,

для которого также может быть создана хранимая процедура. Затем выполняется формирование индексных таблиц *ind*S1, …, *indSp* (p – число S-серверов). Алгоритм бустрофедона работает таким образом, что суммарные значения полей MM во всех индексных таблицах незначительно отличаются друг от друга. Эти индексные таблицы используются на следующем этапе для распределения таблиц.

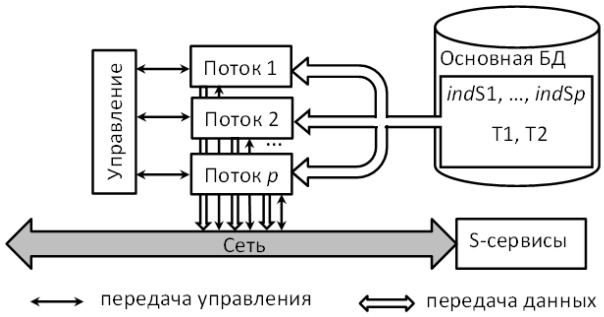

Рис. 3. Процедура распределения таблиц

Процедура распределения таблиц по S-серверам может выполняться параллельно. Количество параллельных потоков определяется возможностями хост-машины, такими как количество процессоров (ядер), и пропускной способностью сети. Существенную роль играют возможности СУБД, обеспечивающие параллельный доступ нескольких запросов к одной таблице. Схема процедуры распределения таблиц приведена на рисунке 3.

Блок управления запускает p потоков, по количеству S-серверов и переходит в режим ожидания их завершения. Потоки взаимодействуют с основной БД и S-сервисами. Они последовательно считывают индексные таблицы и для каждого значения ключа формируют запросы вида

SELECT * FROM T_i WHERE π(<Key>) ORDER BY <Key>, (*)

(i=1, 2), π(<Key>) – предикат, задающий условие отбора строк таблиц T1 и T2. На основе этих запросов потоки последовательно считывают строки таблиц и формируют буферные байт-массивы для каждой таблицы. Размеры байт-массивов определяются типом и пропускной способностью сети, а также они должны быть кратны длине строки таблицы в байтах, что существенно упрощает программирование. Как только очередной буфер заполнен данными, выполняется проверка готовности S-сервиса к приему данных. Если S-сервис готов к приему, запускается асинхронная процедура передачи данных, а поток

возвращается к подготовке очередного буфера, в противном случае поток переходит в режим ожидания готовности S-сервиса. Таким образом, потоки, выполняющиеся на хост-машине, взаимодействуют по управлению с соответствующими им S-сервисами. *j*-поток продолжает выполнять эти действия до тех пор, пока соответствующая ему индексная таблица $indS_j$ не будет исчерпана, и не будет получено сообщение от соответствующего ему S-сервиса о том, что последний отправленный буфер принят, обработан и сохранен в соответствующей таблице фрагмента основной БД. После этого поток возвращает блоку управления сигнал о завершении работы. После того, как все запущенные потоки сообщат о завершении работы, блок управления завершит работу всей программной системы, реализующей симметричное горизонтальное распределение данных.

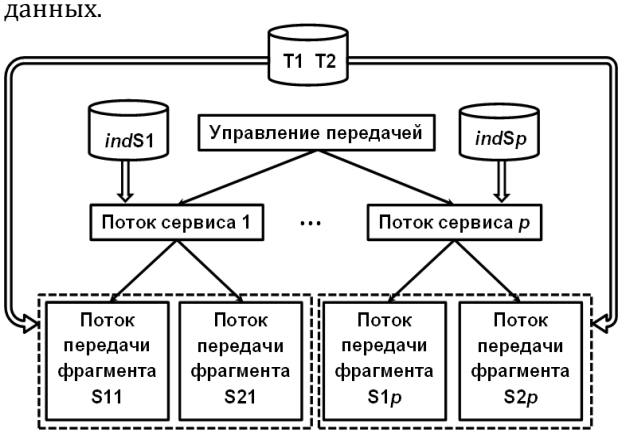

Рис. 4. Параллелизм потоков передачи таблиц

Поскольку чтение таблиц может осуществляться параллельно, каждый поток передачи данных может быть разделен на два независимых параллельно выполняющихся потока. Схема выполнения этих потоков приведена на рисунке 4. На этом рисунке показано, как каждый поток, взаимодействующий с S-сервисом, разделяется на два подчиненных потока. После того, как оба запроса типа (*) сформированы, инициируются два потока, каждый из которых взаимодействует со своей таблицей.

Программное обеспечение S-сервера – S-сервис представляет собой совокупность программ, реализованных как API-контроллеры, обрабатывающие GET и POST запросы от клиента (хост-машины). Реализованы API-контроллеры трех типов, свойства и функции которых приведены в таблице 1.

Далее приводятся подробные описания всех типов API-контроллеров со следующими функциями:

Тип 1. Этот контроллер принимает данные, необходимые для инициализации базы данных.

Возможны три варианта действий, зависящие от того в каком состоянии находится база данных:

- база данных существует, таблицы созданы. В этом случае запрос может содержать только имена таблиц. Контроллер формирует запросы типа DELETE FROM <имя таблицы> и удаляет все данные из таблиц T1 и T2. Тем самым он готовит эти таблицы для заполнения новыми данными. В том случае, когда таблицы содержат большое количество строк, что может привести к большим затратам времени на выполнение операции DELETE FROM, контроллер, воспользовавшись метаданными, удаляет таблицы из БД и создает их заново;
- база данных существует, таблицы не созданы. Запрос содержит имена и схемы таблиц. Контроллер формирует запросы типа CREATE TABLE и выполняет их, создавая таблицы T1 и T2;
- база данных не существует. Запрос содержит имя БД, необходимые для ее создания метаданные и имена и схемы таблиц. Контроллер формирует запрос типа CREATE DATABASE, выполняет его, а затем выполняет все действия, из п. 2.
- Код завершения запроса во всех случаях содержит информацию о том, что запрос завершился успешно, или что при его выполнении возникли ошибки, коды которых входят в общий код завершения.

Таблица 1. API-контроллеры S-сервиса

Тип	Функция контроллера	Тип обрабатываемого запроса	Входные данные	Выходные данные
1	Инициализация фрагмента основной БД	POST	Имена [схемы] таблиц T1 и T2, [схема БД]	Код завершения запроса
2	Формирование фрагмента таблицы T1/T2	POST	Блок данных таблицы T1/ T2	Код завершения запроса
3	Выполнение операции Join	GET	Параметры запроса	Результат выполнения запроса

Тип 2. Прием, распаковка и добавление в БД принятого по сети блока. Такие контроллеры создаются для каждой таблицы-операнда. Это позволяет повысить производительность S-сервиса за счет возможности их параллельного выполнения. Этот контроллер принимает блок данных, в буферный байт-массив. Затем выполняется распаковка этих данных в строки соответствующей таблицы. Полученные строки добавляются в таблицу. Результаты завершения запроса, возвращаемые хост-машине, аналогичны результатам, которые возвращает запрос типа 1.

Тип 3. Выполнение операции Join над таблицами-операндами БД-фрагмента. Это типичный GET-запрос, который может вообще не содержать входных данных. Если хранимая процедура на выполнение операции Join уже есть в БД, запрос запускает ее выполнение, в противном случае он, на основе принятых параметров, формирует запрос к БД и запускает его выполнение. После того, как запрос к БД выполнен, его результаты возвращаются хост-машине таким же способом, как фрагменты таблиц-операндов передаются S-сервису.

По завершении всех действий, связанных с выполнением операции Join, хост-машина принимает результаты от всех S-сервисов и объединяет их в общую таблицу – результат операции Join над таблицами T1 и T2.

При разработке программного обеспечения, реализующего симметричное горизонтальное распределение таблиц, приходится отказываться от применения всех возможностей технологии MVC, в частности, от функций, связанных с моделью данных. Несмотря на то, что эти функции могли бы существенно упростить разработку, при их использовании возникли бы большие потери ресурсов, в частности, времени, из-за необходимости преобразования таблиц-операндов в JSON-формат и обратно и значительного увеличения объема передаваемых по сети данных.

Возможны различные подходы к разработке и использованию предложенной архитектуры программного обеспечения.

Создаются универсальные программы, реализованные в виде библиотечных процедур. Эти программы принимают в качестве параметров схемы БД и таблиц, параметры операции Join, генерируют и передают БД все необходимые запросы, определяют размеры передаваемых блоков таблиц, настраиваются на распаковку и упаковку строк таблиц, определяют допустимое количество параллельных потоков для хост-машины. Достоинство такого подхода состоит в том, он не требует от разработчика больших усилий и существенно сокращает время разработки. Основной, и весьма существенный недостаток состоит в том, что фактически все действия с данными: запросы к БД, упаковка и распаковка строк, выполняются в режиме интерпретации. Это приводит к снижению производительности всего программно-аппаратного комплекса.

Разрабатывается CASE-средство, которое на основе технологии паттернов генерирует программное обеспечение для хост-машины и S-сервисов. Это программное обеспечение включает в себя хранимые процедуры для выполнения всех пользовательских запросов к БД. Программное обеспечение узлов сети может быть настроено на их физические характеристики и на характеристики используемой сети. Естественно, в этом случае разработка программного обеспечение потребует больших временных затрат и более высокой квалификации разработчика.

Для оценки предложенного метода был проведен вычислительный эксперимент. Разработанная архитектура была реализована на локальной сети. Хост-машина имела процессор Intel Core i7 (8 виртуальных ядер), емкость оперативной памяти – 16 Гбайт и емкость жесткого диска – 1 Тбайт. В качестве S-серверов были использованы вычислительные системы с четырех ядерными процессорами Intel Core i5, емкостью оперативной памяти – 8 Гбайт и емкостью жесткого диска – 512 Гбайт. Скорость передачи данных по сети была специально выбрана невысокой – 100 Мбит/сек., что позволило приблизить условия эксперимента к возможности использования сети интернет. Обработке подвергались таблицы T1 и T2, с одинаковым количеством строк в каждой. Число строк изменялось от 100 тысяч до 1 миллиона 100 тысяч. Результаты эксперимента приведены в таблице 2.

Таблица 2. Зависимость симметричного горизонтального распределения данных от объема таблиц

Объем таблицы	Время подготовки данных	Время передачи данных
100	00:00,5	00:19,5
200	00:01,5	00:26,1
300	00:01,8	00:25,6
400	00:02,3	00:27,5
500	00:02,6	00:31,3
600	00:02,9	00:42,5
700	00:04,4	00:44,3
800	00:05,5	00:53,0
900	00:06,3	00:58,8
1000	00:07,0	01:02,2
1100	00:07,2	01:17,1

Эти же результаты в графической форме приведены на рисунках 5 и 6.

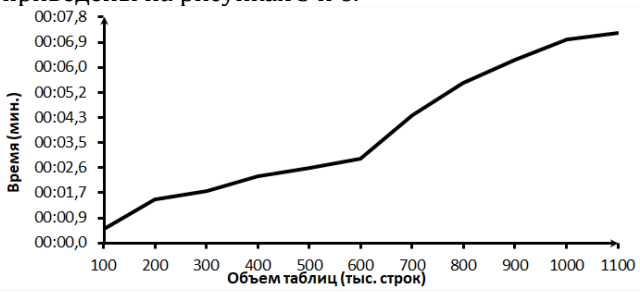

Рис. 5. Зависимость времени подготовки данных от объема таблиц

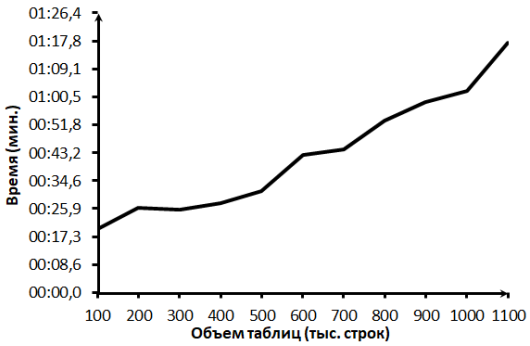

Рис. 6. Зависимость времени передачи данных от объема таблиц

Из приведенных результатов можно сделать вывод о том, что время реализации симметричного горизонтального распределения практически линейно зависит от объемов исходных данных.

При сравнении с временем обработки заранее распределенных таблиц при тех же условиях [4, 5], можно утверждать, что время симметричного горизонтального распределения двух таблиц-операндов большого объема незначительно (не более чем на 5%) увеличивает время параллельного выполнения операции Join, которое в разы меньше времени ее последовательного выполнения.

В заключение следует отметить, что подобная (MPP) архитектура программно-аппаратного комплекса дает наилучшие результаты в том случае, когда таблицы-операнды имеют очень большие объемы, высокие коэффициенты активности. Тогда затраты на симметричное горизонтальное распределение таблиц позволит при большом числе S-серверов получить незначительные объемы фрагментов основной БД, что существенно ускорит выполнение операции Join.

Литература

1. Мунерман В.И. Построение архитектур программно-аппаратных комплексов для повышения эффективности массовой обработки данных. – Системы высокой доступности, № 4, 2014, т.10, с. 3-16.
2. Martello S., Toth P. Knapsack problems: algorithms and computer implementations. – John Wiley & Sons, Inc. New York, NY, USA ©1990 – p. 306, ISBN:0-471-92420-2.
3. Клири С. Введение в async/await в ASP.NET. – https://msdn.microsoft.com/ru-ru/magazine/dn802603.aspx.
4. Мунерман В.И. Реализация обработки больших объемов данных на симметричных мультипроцессорных системах. – Системы высокой доступности. 2013. Т. 9. № 2. – С. 036-039.
5. Данилюк Е.И., Мунерман В.И. Сравнительный анализ реализации операции Join на комплексах с SMP и MPP архитектурой. – Системы компьютерной математики и их приложения: материалы XVIII Международной научно-практической конференции. – Смоленск, Изд-во СмолГУ, 2017. – Вып. 18. – С.66-68.

References

1. Munerman V.I. Postroenie arhitektur programmno-apparatnyh kompleksov dlja povyshenija jeffek-tivnosti massovoj obrabotki dannyh. – Sistemy vysokoj dostupnosti, № 4, 2014, t.10, s. 3-16.
2. Martello S., Toth P. Knapsack problems: algorithms and computer implementations. – John Wiley & Sons, Inc. New York, NY, USA ©1990 – p. 306, ISBN:0-471-92420-2.
3. Kliri S. Vvedenie v async/await v ASP.NET. – https://msdn.microsoft.com/ru-ru/magazine/dn802603.aspx.
4. Munerman V.I. Realizacija obrabotki bol'shih ob#emov dannyh na simmetrichnyh mul'tiprocessornyh sistemah. – Sistemy vysokoj dostupnosti. 2013. T. 9. № 2. – S. 036-039.
5. Danilyuk E.I.. Munerman V.I. Sravnitelnyy analiz realizatsii operatsii Join na kompleksakh s SMP i MPP arkhitekturoy. – Sistemy kompyuternoy matematiki i ikh prilozheniya: materialy XVIII Mezhdunarodnoy nauchno-prakticheskoy konferentsii. – Smolensk. Izd-vo SmolGU. 2017. – Vyp. 18. – S.66-68.

Поступила: 15.09.2017

Об авторах:

Мунерман Виктор Иосифович, кандидат технических наук, доцент кафедры информатики, Смоленский государственный университет, vimoon@gmail.com

Мунерман Даниил Викторович, лаборант-стажер кафедры информатики, Смоленский государственный университет, danvmoon@gmail.com

Note on the authors:

Munerman Victor I., Candidate of Technical Sciences, associate professor "Information technologie", Smolensk State Unversity, vimoom@gmail.com

Munerman Daniel V., laboratory assistant "Information technologie", Smolensk State University, danvmoon@gmail.com