УДК 004.41

Надэлин А.А.

Смоленский государственный университет, г. Смоленск, Россия

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ОПЕРАЦИИ JOIN СРЕДСТВАМИ ТЕХНОЛОГИИ CUDA

АННОТАЦИЯ

В работе рассматривается метод параллельной реализации операции Join. Для решения проблемы используется принцип симметричного горизонтального распределения данных и многоядерные графические процессоры с использованием технологии CUDA. Приведены результаты эксперимента, который показал преимущество представленного подхода.

КЛЮЧЕВЫЕ СЛОВА

Базы данных; графические процессоры; принцип симметричного горизонтального распределения данных; технология CUDA.

Nadelin A.A.

Smolensk State University, Smolensk, Russia

PARALLEL IMPLEMENTATION OF THE OPERATION JOIN BY MEANS OF TECHNOLOGY CUDA

ABSTRACT

Method of parallel implementation of the operation join review in this paper. To solve the problem used principle of symmetrical horizontal data distribution and multicore GPUs with CUDA technology. An experiment which confirms the advantage of the proposed approach is given in the article.

KEYWORDS

Database; GPUs; the principle of horizontal symmetrical distribution of data; CUDA technology.

Повышение производительности систем управления базами данных (СУБД) – одна из самых актуальных задач, направленных на ускорение обработки структурированных больших данных. Одна из проблем, решение которой позволит существенно увеличить производительность СУБД, заключается в том, что одна из реляционных операций – операция соединения (JOIN) – относится к числу "медленных" операций. Действительно, алгоритм, реализующий операцию JOIN, имеет вычислительную сложность пропорциональную произведению числа строк в таблицах, следовательно, время выполнения операции тоже будет пропорционально этой величине. Если таблицы имеют большие объемы (десятки миллионов – миллиарды строк), время выполнения этой операции перестает удовлетворять пользователей.

Один из подходов, направленных на повышение производительности СУБД, состоит в, так называемой, денормализации базы данных. Он предполагает намеренное приведение структуры базы данных в состояние, не соответствующее критериям нормализации. Обычно денормализация проводится с целью ускорения операций чтения данных из базы за счет добавления избыточных данных. Полагается, что введение избыточности позволит увеличить производительность. Однако, такой подход порождает проблемы, связанные с доступом к данным при их корректировке и может, в силу значительного увеличения объемов данных, существенно снизить производительность запросов, особенно в тех случаях, когда эти запросы должны обработать значительную часть данных.

Другой подход состоит в использовании технологий in-memory. Однако, когда объемы данных чрезвычайно велики, использование этих технологий на всех данных может оказаться не только затруднительным, но и невозможным. Вместе с тем элементы этого подхода могут использоваться, и были использованы, при реализации предложенного в настоящей работе метода повышения производительности СУБД при выполнении операции JOIN.

Уменьшить время выполнения операции JOIN можно за счет параллельной обработки не

связанных между собой элементов таблиц данных, на разных процессорах или ядрах процессора. Далее рассматривается метод распараллеливания операции JOIN, основанный на использовании технологии CUDA [1, 2]и принципа симметричного горизонтального распределения данных [3].

Технология CUDA – это программно-аппаратная архитектура, позволяющая производить на графическом процессоре вычисления общего назначения. При этом графический процессор играет роль мощного сопроцессора. В данное время технология CUDA позволяет создавать программное обеспечение для решения сложных вычислительных задач за меньшее время. Происходит это в результате многоядерной вычислительной мощности графических процессоров. Количество CUDA ядер в составе графического процессора может достигать нескольких тысяч, что позволяет существенно распараллелить процесс вычисления. Технология CUDA также используется для повышения производительности при реализации запросов к базам данных.

Принцип симметричного горизонтального распределения данных, обеспечивающий возможность параллельной реализации операции Join, позволяет в сочетании с технологией CUDA эффективно реализовать алгоритм, основанный на вычерпывании [4, 5] ведущей таблицы. Пусть таблицы X_K (ведущая) и Y_K (ведомая) нестрого упорядочены по множеству ключей K. В реляционной терминологии это означает, что они находятся в первой нормальной форме, то есть могут содержать группу строк с одинаковым значением не первичного ключа K (в общем случае составного). Такой ключ обычно задается в параметре ON операции Join. Без ограничения общности можно считать, что обе таблицы состоят из классов эквивалентности, соответствующих одним и тем же значениям экземпляров ключа K. Тогда эти таблицы могут быть представлены как фактор-множества $X_K = \{X_{K_1^*}, ..., X_{K_p^*}\}$ и $Y_K = \{Y_{K_1^*}, ..., Y_{K_p^*}\}$ где классы эквивалентности $X_{K_i^*}$ и $Y_{K_i^*}$ есть совокупности $X_{K_i^*} = \{x_{i1}, ..., x_{iq_i}\}$ и $Y_{K_i^*} = \{y_{i1}, ..., y_{ir_i}\}$ строк этих таблиц. Особенность рассматриваемого алгоритма состоит в том, что каждый класс эквивалентности таблиц X_K и Y_K целиком считывается (зачерпывается) в буферную область оперативной памяти (черпак). Если экземпляр ключа хранящихся в черпаке строк ведущей таблицы, совпадает с экземпляром ключа хранящихся в черпаке строк ведомой таблицы, производится совместная обработка всех строк обоих черпаков. В результате для каждого экземпляра ключа формируется декартово произведение соответствующих ему классов эквивалентности таблиц X_K и Y_K. Для реализации предложенного параллельного алгоритма операции Join предложена архитектура программно-аппаратного комплекса, которая основана на использовании графических процессоров (рисунок 1).

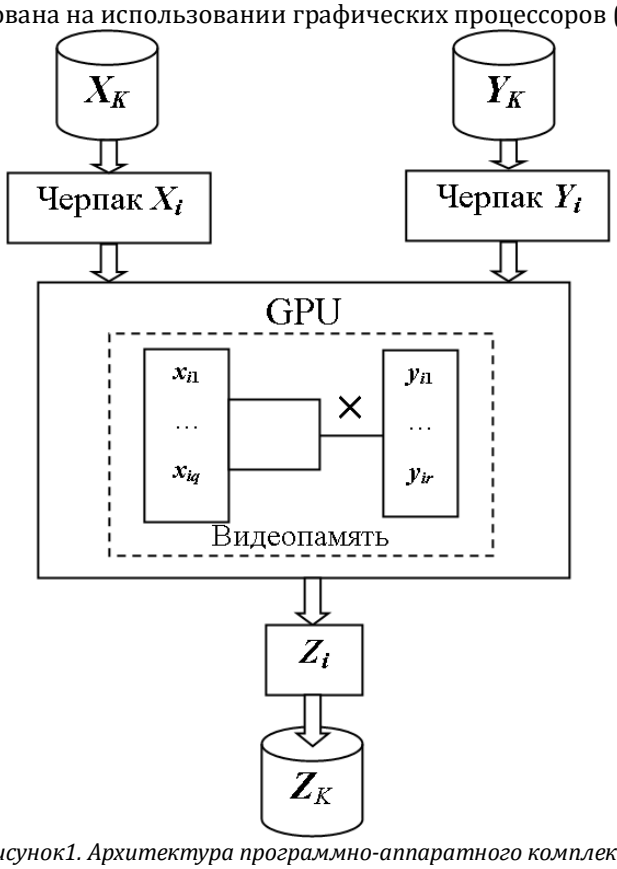


Рисунок1. Архитектура программно-аппаратного комплекса

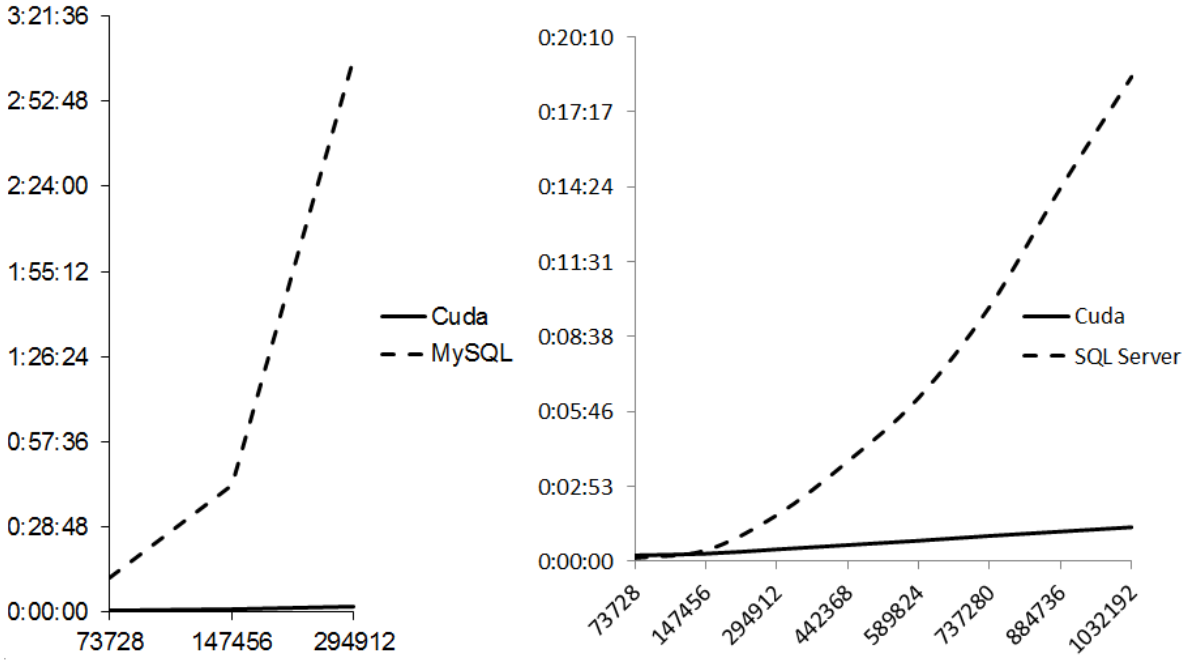
Для проверки результатов распараллеливания операции Join был проведен эксперимент с использованием двух вычислительных архитектур. В первом случае операция Join была реализована над двумя таблицами с помощью предложенного алгоритма. Во втором случае операция JOIN тех же таблиц осуществлялась средствами СУБД, то есть выполнялся запрос

SELECT T1.K, SUM(T1.V*T2.V) FROM T1 INNER JOIN T2 ON T1.K = T2.K GROUP BY T1.K ORDER BY T1.K

В запросе обрабатываются две таблицы со схемами T1(K, V) и T2(K, V). Поле K – это ключ, идентифицирующий строки таблиц в параметре ON, поле V – произвольное числовое поле.

Эксперимент поводился с использование двух СУБД: MySQL и Microsoft SQL Server 2014. Выбор этих СУБД определился тем, что они существенно различаются по производительности, кроме того СУБД Microsoft SQL Server 2014 многие действия, в том числе чтение таблиц и операцию Join реализует параллельно.

Для эксперимента была использована 384-ядерная видеокарта Gigabyte GT730. Выполнялась операция JOIN над парами таблиц с одинаковым числом строк, которое изменялось от 73728 до 442368 (кратно числу ядер графического процессора). Программа, с помощью которой осуществлялся эксперимент, написана в среде программирования Intellij IDEA на языке программирования java с использованием драйвера jtds-1.3.1. Среда разработки Intellij IDEA , язык программирования java и драйвер jtds-1.3.1 были выбраны из-за сравнительно простой и быстрой организации передачи данных между базой данных находящейся на жестком диске и видеопамятью графического процессора. Для организации запросов Select, которые используются для получения выборок из базы данных, использовались хранимые процедуры SQL, это позволило повысить быстродействие системы. Результаты, полученные в ходе эксперимента, приведены на рисунке 2.



А) Б)

Рис. 2. Сравнение времени выполнения операции JOIN средствами СУБД и технологии CUDA

В ходе эксперимента был проверен случай, когда оба класса эквивалентности помещаются в память графического процессора, но возможны два частных случая:

1. Один класс эквивалентности помещается в память графического процессора полностью, а второй частично. В этом случае второй класс эквивалентности разбивается на куски, достаточно малые для того чтобы поместиться в память, затем поочередно находится декартово произведение с частями второго класса эквивалентности. Совокупность этих декартовых произведений и будет результирующим декартовым произведением.
2. Ни один из классов эквивалентности ни помещается в память графического процессора целиком. В этом случае оба класса эквивалентности разбиваются на части, и затем производится декартово произведение с ними поочередно.

Таким образом, на основании результатов эксперимента можно сделать вывод о том, что предложенный алгоритм параллельной реализации операции JOIN средствами технологии существенно превосходить по быстродействию реализацию этой операции в СУБД.

Наилучшие результаты применения технологии CUDA можно получить уменьшив до минимума объем данных передаваемых из памяти центрального процессора в память графического процессора и наоборот из памяти графического процессора в память центрального процессора. В предыдущем эксперименте была применена архитектура, при которой объем данных, передаваемых в память центрального процессора, из памяти графического процессора был минимальным. Целью второго эксперимента является определение целесообразности использования графического процессора, при вычислении декартова произведения без последующего сложения. То есть выполняемый запрос примет вид:

SELECT FROM T1 INNER JOIN T2 ON T1.K = T2.K GROUP BY T1.K ORDER BY T1.K

По данным результатов эксперимента построен график (рисунок 3).

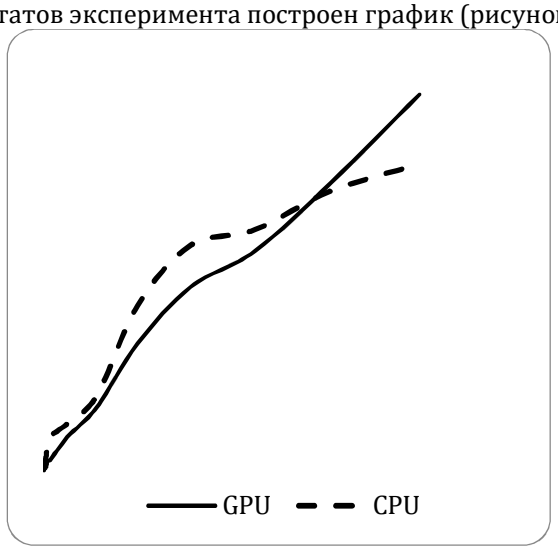


Рисунок 3. Результаты эксперимента

Из графика видна зависимость затрат времени от обьема данных, передаваемых из памяти графического процессора в память центрального процессора. При небольших обьемах классов эквивалентности алгоритм , использующий технологию CUDA работает быстрее, чем стандартная СУБД, но при увеличении классов эквивалентности, затраты времени на передачу данных превосходят разницу во времени выполнения действий над данными и использование технологии CUDA становится не желательным.

В ходе работы был разработан метод, позволяющий существенно повысить скорость выполнения операции Join в базах данных большого объема за счет использования принципа симметричного горизонтального распределения базы данных и применения технологии CUDA. У этого метода есть свои особенности:

1. Метод эфиктивет только при достаточно больших размерах входных классов эквивалентности. Потому что при малых размерах входных классов эквивалентности время на обработку данных центральным процессором меньше времени передачи данных в видеопамять.
2. Метод эфективен только при малых размерах выходного класса эквивалентности (это видно во втором эксперименте). Поэтому не целесообразно передавать декартово произведение из видеопамяти. Целесообразно использовать метод тогда когда находится сумма декартова произведения. Это происходит из-за затрат времени на передачу данных из памяти графического процессора в оперативную память.
3. При вычисленнии операции Join оба входных класса эквивалентности должны помещаться в память графического процессора.
4. Существуют особенности применения метода когда входные классы эквивалентности не помещаются целиком в память графического процессора. В этом случае входные классы эквивалентности должны разбиваться на части и над этими частями должны проводиться соответствующие действия.

С учетом всех этих особенностей можно сделать вывод о том что данный метод необходимо встраивать в систему управления базами данных а не использовать отдельно. Система управления базами данных должан выбирать метод выполнения операции JOIN в зависимости от размеров входных и выходных классов эквивалентности. При больших размерах входных и малых размерах выходных классов эквивалентности должен запускаться алгоритм использования графического процессора, в остальных случаях должен использоваться центральный процессор. Если учесть все

эти особенности, то система управления базами данных с использование принципа симметричного горизонтального распределения данных и технологии CUDA дадут существенный выйгрыш во времени по сравнению с использованием стандартных систем управления базами данных.

Работа выполнена под руководством кандидата технических наук доцента Мунермана В. И.

Литература

1. Боресков А. В., Предисл.: Садовничий В. А. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учебное пособие. **–** Издательство Московского университета, 2012. – 336 стр., ISBN: 978-5-211-06340-2.
2. Bakkum P., Skadron K. Accelerating SQL database operations on a GPU with CUDA. – Proceedings of the 3rd Workshop on GeneralPurpose Computation on Graphics Processing Units. ACM, 2010. – pp. 94–103.
3. Н.А. Левин, В.И. Мунерман Модели обработки больших объемов данных в системах массового параллелизма. – М.: Системы высокой доступности, 1, 2013, v. 9. – с. 35-43.
4. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.
5. В.И.Мунерман В.И.Мунерман Параллельная реализация операции слияния нестрого упорядоченных файлов. Смоленский государственный университет.

References

1. Boreskov A. V., Predisl.: Sadovnichiy V. A. Parallel'nye vychisleniya na GPU. Arkhitektura i programmnaya model' CUDA: Uchebnoe posobie. – Izdatel'stvo Moskovskogo universiteta, 2012. – 336 str., ISBN: 978-5-211-06340-2.
2. Bakkum P. , Skadron K. Accelerating SQL database operations on a GPU with CUDA. – Proceedings of the 3rd Workshop on GeneralPurpose Computation on Graphics Processing Units. ACM, 2010. – pp. 94–103.
3. N.A. Levin, V.I. Munerman Modeli obrabotki bol'shikh ob"emov dannykh v sistemakh massovogo parallelizma. – M.: Sistemy vysokoy dostupnosti, 1, 2013, v. 9. – s. 35-43.
4. Akho A., Khopkroft Dzh., Ul'man Dzh. Postroenie i analiz vychislitel'nykh algoritmov. – M.: Mir, 1979. – 536 s.
5. V.I.Munerman, V.I.Munerman Parallel'naya realizatsiya operatsii sliyaniya nestrogo uporyadochennykh faylov .Smolenskiy gosudarstvennyy universitet.

Поступила: 12.09.2016

Об авторах:

Надэлин Александр Александрович, магистрант Смоленского государственного университета, sanya.nadelin@yandex.ru.