РАЗРАБОТКА МЕТОДОВ ОБРАБОТКИ ЗАПРОСОВ К БАЗЕ ДАННЫХ НА МНОГОЯДЕРНЫХ УСКОРИТЕЛЯХ, ПОДДЕРЖИВАЮЩИХ ТЕХНОЛОГИЮ CUDA

П.С. Костенецкий, А.И. Семенов

1. Введение

После появления технологии NVIDIA CUDA, позволяющей проводить произвольные вычисления на графических процессорах, возникло новое направление научных исследований, посвященное обработке баз данных с использованием графических ускорителей [2]. В данном направлении известны работы, посвященные ускорению обработки запросов к базе данных [1, 5] и оптимизации процесса интеллектуального анализа данных (data mining) [3, 4]. Так как пропускная способность видеопамяти примерно в 3 раза превышает среднюю скорость работы с оперативной памятью, возникает вопрос эффективности не только обработки, но и размещения базы данных непосредственно в графической памяти. На сегодняшний день на рынке доступны гибридные вычислительные серверы с объемом видеопамяти, достигающим 24 Гб. Таких объемов памяти обычно достаточно для эффективного хранения и обработки большинства баз данных. Для увеличения объема хранимой в видеопамяти базы данных, отдельные серверы могут быть объединены в вычислительные кластеры. Кроме того, могут быть реализованы механизмы загрузки блоков данных из оперативной памяти вычислительных узлов.

Данная работа посвящена оценке эффективности хранения и обработки баз данных непосредственно в графической памяти видеокарт с поддержкой технологии программирования CUDA, а также поиску и реализации операций над базами данных, для выполнения которых применение графических процессоров является эффективным. Для выполнения исследований авторами был реализован эмулятор СУБД, моделирующий параллельную обработку запросов [6] на графическом ускорителе, а так же обработку этих же запросов на многоядерном центральном процессоре. При помощи эмулятора исследована эффективность использования технологии CUDA для хранения и обработки баз данных на графических ускорителях.

2. Методы реализации запросов к базе данных

В ходе исследования были разработаны методы обработки наиболее распространенных типов запросов JOIN и SELECT. Методы были реализованы в виде программного эмулятора СУБД, позволяющего моделировать параллельное выполнение запросов к базе данных на многоядерных ускорителях, поддерживающих технологию CUDA и на многоядерном центральном процессоре.

Реализация запроса SELECT

SELECT – оператор языка SQL, возвращающий набор данных (выборку) из базы данных, удовлетворяющих заданному условию. При формировании запроса SELECT описывается ожидаемый набор данных: его вид (набор атрибутов) и его содержимое (критерий попадания записи в набор, группировка значений, порядок вывода записей и т.п.). Запрос выполняется следующим образом: сначала извлекаются все

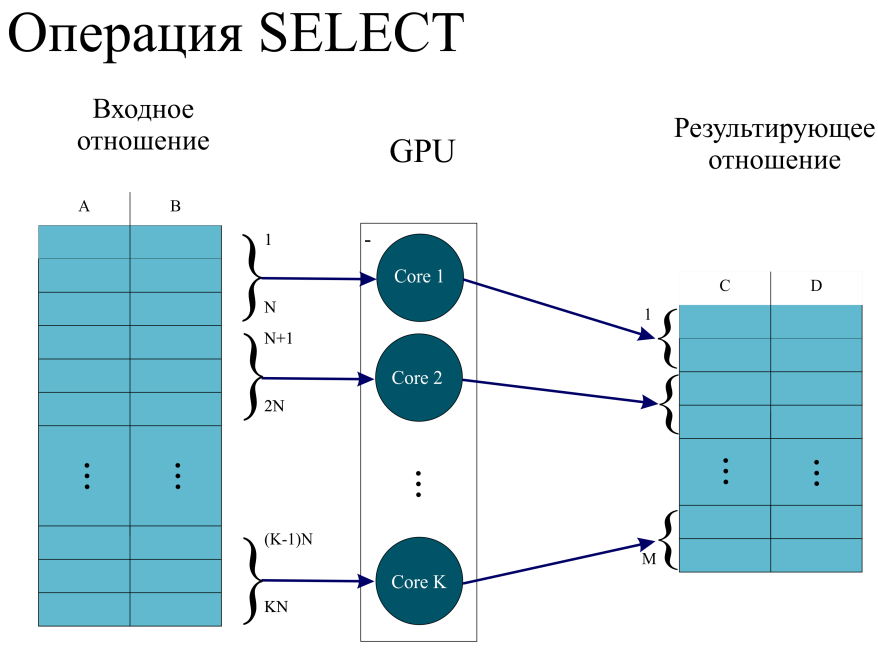

Рис. 1. Схема выполнения операции SELECT

записи из отношения, а, затем, для каждой записи набора проверяется ее соответствие заданному критерию. На рис. 1. приведена схема предлагаемого метода обработки запроса SELECT на графическом ускорителе.

Алгоритм выполнения запроса SELECT на графическом ускорителе:

1) генерация входного отношения R в оперативной памяти;
2) выделение памяти графического ускорителя под массив для результатов;
3) копирование отношения R из оперативной памяти в память графического ускорителя;
4) логическое разбиение отношения R на фрагменты (количество фрагментов равно количеству нитей CUDA);
5) выполнение выборки из отношения R на графическом ускорителе (каждая нить обрабатывает один фрагмент, запись результатов проводится в созданный ранее в памяти графического ускорителя массив);
6) копирование результата из памяти графического ускорителя в оперативную память.

Алгоритм выполнения запроса SELECT на многоядерном центральном процессоре:

1) генерация входного отношения R;
2) выполнение выборки из отношения R (нити OpenMP в цикле обрабатывают кортежи и записывают результаты в результирующий динамический массив);

Реализация запроса JOIN

Был реализован запрос INNER JOIN, выполняющий внутреннее соединение двух отношений методом вложенных циклов. Заголовок результирующего отношения является объединением заголовков соединяемых отношений. Результирующее отношение формируется следующим образом. Каждый кортеж одного отношения сопоставляется с каждым кортежем второго отношения, после чего проверяется условие соединения (вычисляется предикат соединения). Если условие истинно, кортежи добавляется в результирующее отношение. На рис. 2. приведена схема предлагаемого метода обработки запроса JOIN на графическом ускорителе.

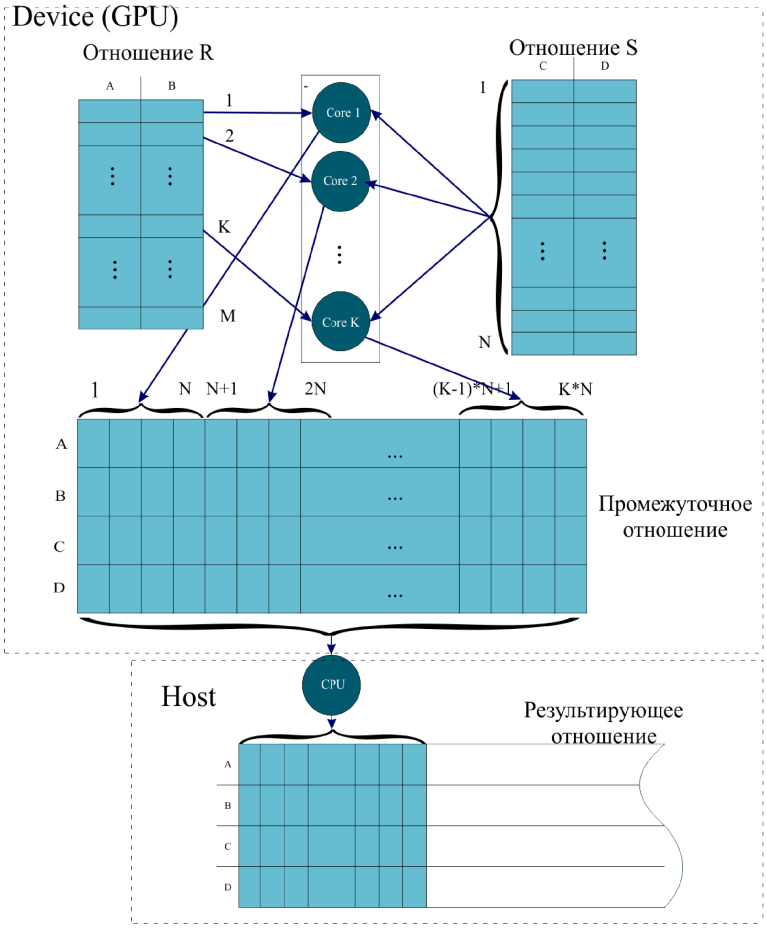

Рис. 2. Схема выполнения операции JOIN

Алгоритм выполнения запроса INNER JOIN на графическом ускорителе:

1)генерация входных отношений R и S в оперативной памяти;
2)выделение памяти графического ускорителя под массив для результатов;
3)копирование отношений R и S из оперативной памяти в память графического ускорителя;
4)логическое разбиение отношения R на фрагменты (количество кортежей во фрагменте равно количеству нитей CUDA);
5)выполнение соединения фрагмента отношения R и отношения S на графическом ускорителе (каждая нить обрабатывает один кортеж R, запись результатов проводится в созданный ранее в памяти графического ускорителя массив);
6)копирование промежуточного результата из памяти графического ускорителя в оперативную память;
7)копирование промежуточного результата в результирующее отношение;
8)очистка массива для промежуточных результатов;
9)повторение пунктов 5-9 для следующего фрагмента отношения R.

Алгоритм выполнения запроса JOIN на многоядерном центральном процессоре:

1) генерация входных отношений R и S;

2) выполнение соединения отношений R и S (нити OpenMP в цикле обрабатывают кортежи и записывают результаты в результирующий динамический массив);

3. Вычислительные эксперименты

Вычислительные эксперименты проводились с использованием разработанного эмулятора СУБД на оборудовании с характеристиками, приведенными в Табл. 1.

Табл. 1. Характеристики используемого оборудования

Оборудование	Характеристики	
Процессор	Intel Core i7-2600 3,8 ГГц	
ОЗУ	8 Гб DDR3-1333 (пропускная способность 10.7 Гб/с)	
Жесткий диск	1 Тб SATA 2	
Системная шина	PCI Express 2.0 16X (пропускная способность 8 Гб/с)	
Графический ускоритель	Модель	NVIDIA GeForce GTX 550 Ti
	CUDA Cores	192
	Тактовая частота памяти	4100 МГц
	Объем памяти	1024 Мб GDDR5
	Разрядность памяти	192-bit
	Пропускная способность памяти	98.4 Гб/с

Для проведения вычислительных экспериментов было сгенерировано 5 тестовых баз данных, состоящих из двух отношений *R* и *S*. Размеры отношений для операции соединения представлены в Табл. 2, для операции выборки – в Табл. 3. Отношения состоят из двух атрибутов и содержат данные типа *Integer*.

Табл. 2. Размеры отношений для операции соединения

№ тестовой БД	Размер *R* (кортежей)	Размер *S* (кортежей)
1	200	20000
2	400	40000
3	600	60000
4	800	80000
5	1000	100000

Табл. 3. Размеры отношений для операции выборки

№ тестовой БД	Размер входного отношения (кортежей)
1	15000000
2	30000000
3	45000000
4	60000000
5	75000000

Исследование эффективности операции соединения для GPU

В данном пункте приводятся графики ускорения обработки запросов на графическом ускорителе.

На Рис. 3 приведена зависимость ускорения выполнения запроса INNER JOIN над базой данных №5 (см. Табл. 2) на GPU при варьирующемся от 32 до 384 количестве нитей CUDA. Следует отметить, что используемый графический ускоритель имеет 192 физических ядра. Как видно из графика, при увеличении количества нитей свыше 192, ускорение начинает уменьшаться. Это объясняется тем, что одновременно на 1 ядре может выполняться только одна нить. Если нитей больше, чем физических ядер, то выполнение нитей происходит в режиме разделения времени.

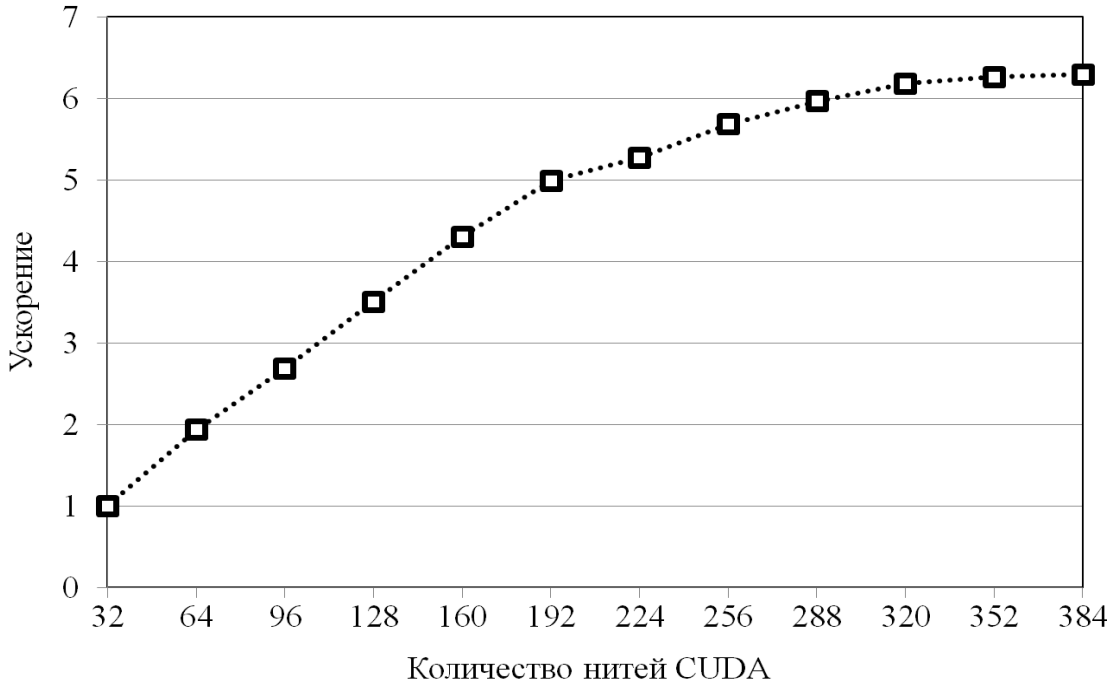

Рис. 3. Ускорение обработки запроса INNER JOIN на GPU

На Рис. 4 приведено время выполнения запроса INNER JOIN на графическом ускорителе при варьирующемся количестве кортежей. Из графика видно, что использование количества нитей, превышающего количество физических ядер более чем на одну треть, не вызывает существенного ускорения обработки запроса.

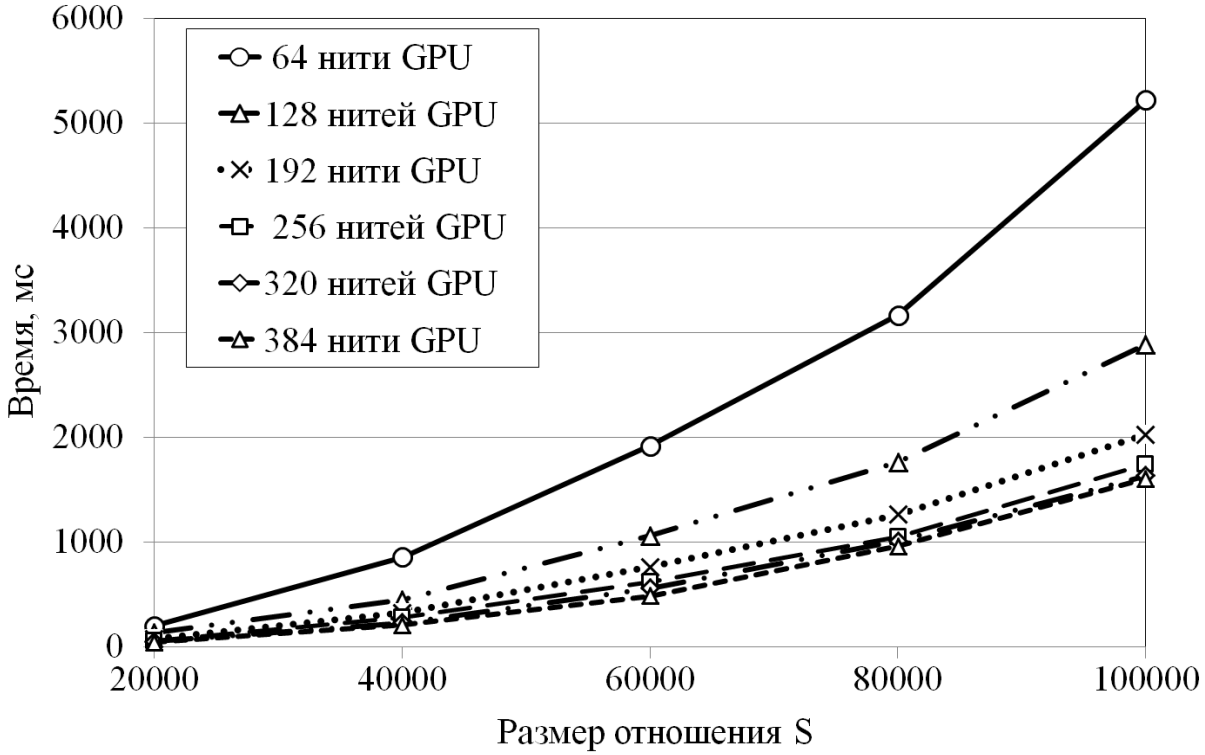

Рис. 4. Время выполнения запроса INNER JOIN на GPU

Исследование эффективности соединения для CPU

Для сравнения производительности различных аппаратных архитектур на Рис. 5 и Рис. 6 приводятся графики ускорения обработки запросов на центральном процессоре при варьирующемся количестве нитей OpenMP.

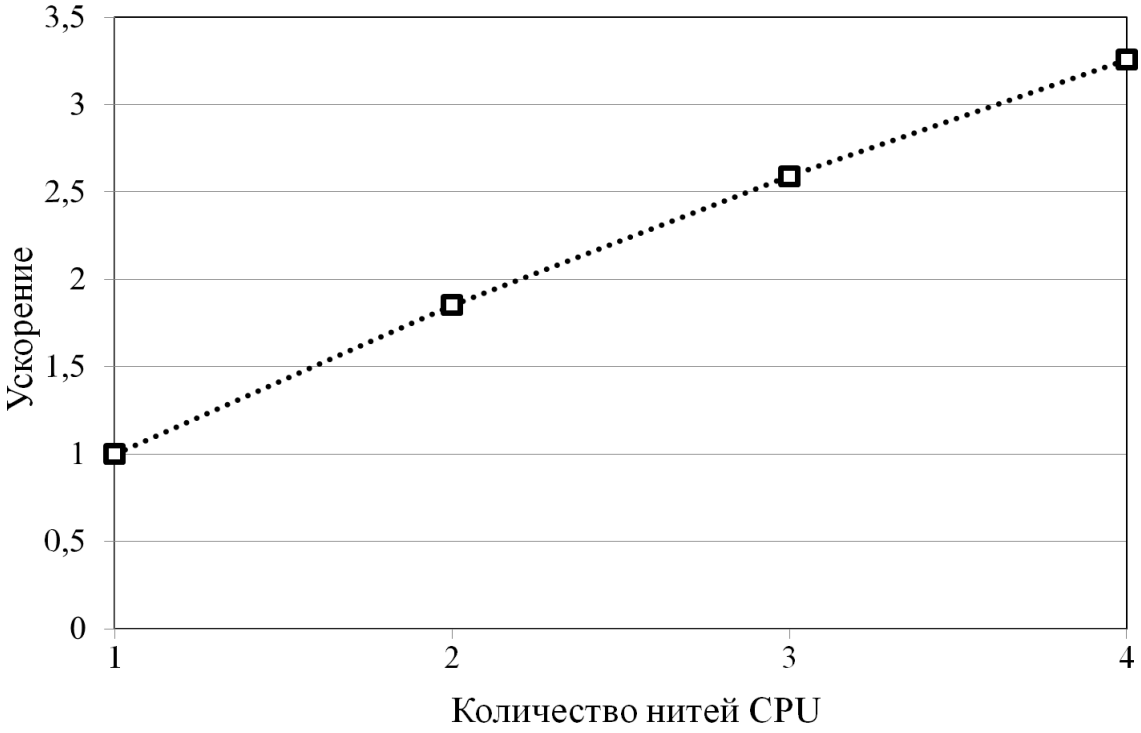

Рис. 5. Ускорение обработки тестовой базы данных №5 на CPU

При реализации алгоритма JOIN на CPU использование количества нитей, превосходящего количество физических ядер, приводит к значительному снижению производительности, в отличие от реализации на CUDA.

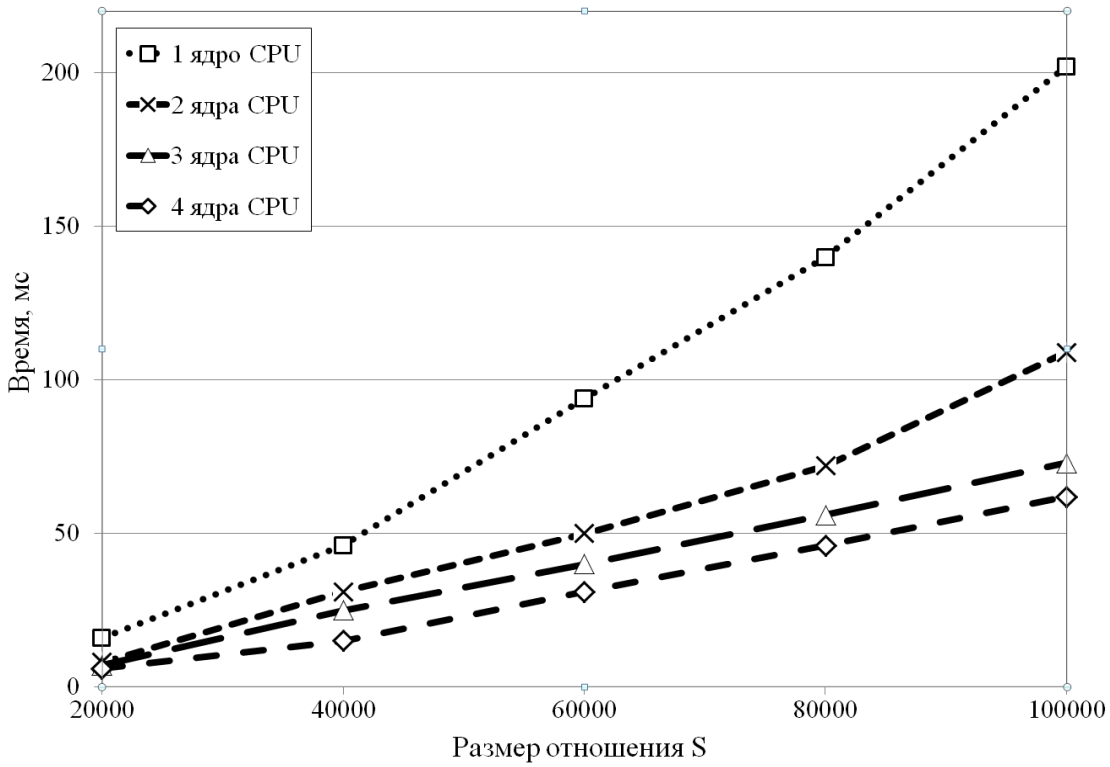

Рис. 6. Время выполнения запроса INNER JOIN на CPU

Из графиков видно, что использование CPU дает достаточно высокое ускорение. Например, использование четырех ядер CPU дает ускорение более чем в три раза. На основании данных экспериментов можно сделать вывод, что для операции JOIN при реализации вложенными циклами эффективнее использовать CPU, чем GPU. Это связано с низкой пропускной способностью шины PCI Express. Например, пропускная способность шины PCI Express 3.0, используемой в большинстве современных графических ускорителей, не

превышает 32 ГБ/с, что накладывает серьезные ограничения на скорость обмена данными между GPU и оперативной памятью.

Исследование эффективности операции выборки для GPU

Как видно из графика ускорения обработки тестовой базы данных №5 на GPU при варьирующемся количестве нитей CUDA (см. Рис. 7), параллельная реализация операции SELECT на CUDA дает ускорение в 4,3 раза при использовании 192 нитей по сравнению с обработкой на 32 нитях.

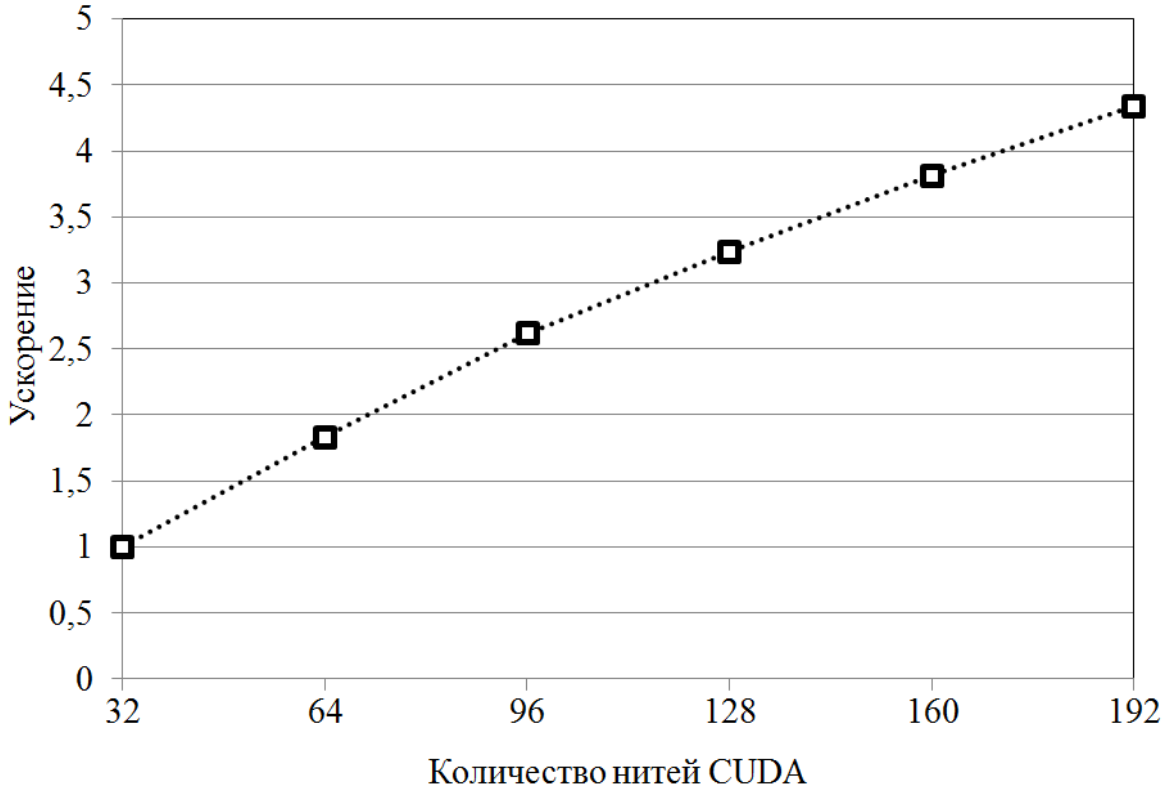

Рис. 7. Ускорение обработки тестовой базы данных №5 на GPU

Исследование эффективности операции выборки для CPU

Из графика ускорения обработки тестовой базы данных №5 на CPU при варьирующемся количестве нитей OpenMP, представленного на Рис. 8, видно, что для операции SELECT увеличение количества ядер приводит к небольшому приросту производительности. К примеру, использование четырех ядер создает ускорение в 1.3 раза по сравнению со временем обработки на одном ядре. После анализа графиков времени и ускорения можно сделать вывод, что использование CPU гораздо менее эффективно для данного типа запроса, чем использование GPU.

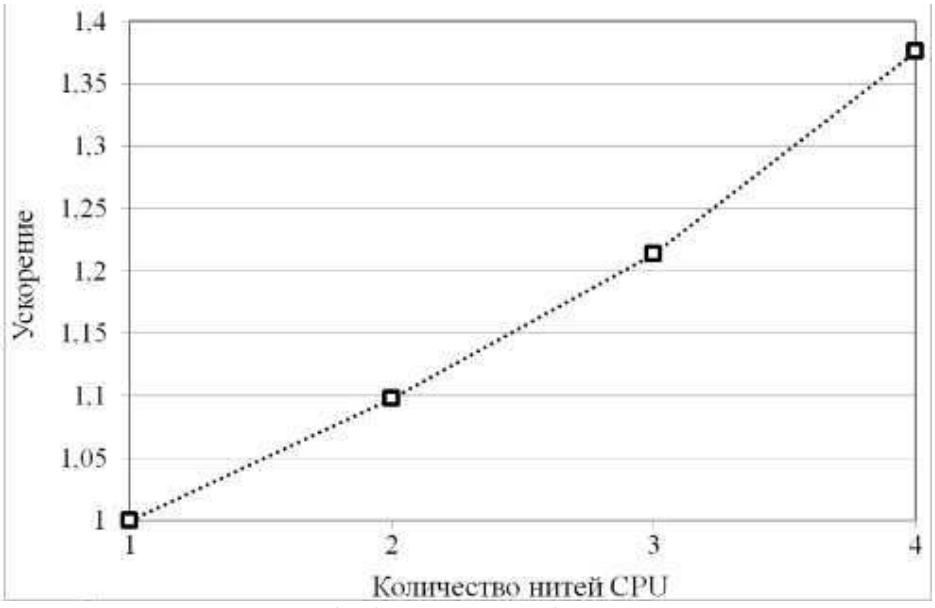

Рис. 8. Ускорение обработки тестовой базы данных №5 на CPU

Если сравнивать производительность CPU и GPU, то для запроса SELECT использование CUDA является более эффективным. При помощи технологии CUDA на бюджетной видеокарте GeForce GTX 550 Ti достигнута эффективность ускорения работы запроса SELECT в 33 раза по сравнению с обработкой на четырех ядрах CPU Intel Core i7.

4. Заключение

В рамках данной работы был реализован эмулятор СУБД, моделирующий обработку запросов к базе данных SELECT и JOIN с использованием технологий обработки запросов в параллельных СУБД на ускорителях, обеспечивающих поддержку технологии CUDA. Установлено, что параллельная обработка при помощи GPU операций над базой данных COUNT, MIN, MAX, SUM, AVG является эффективной. Проведены эксперименты, посвященные обработке запросов SELECT и JOIN на графических процессорах. Установлено, что для операции JOIN использование технологии CUDA неэффективно. В то же время, для запроса SELECT достигнута эффективность ускорения обработки запроса в 33 раза по сравнению с обработкой на центральном процессоре.

Направлением дальнейших исследований будет реализация и тестирование разработанных методов обработки баз данных на многоядерных сопроцессорах Intel MIC. Предполагается, что при обработке больших баз данных архитектура Intel MIC покажет гораздо лучшие результаты за счет наличия когерентного кэша L2, что позволит увеличить производительность при работе с потоками данных, в частности, при выполнении запроса JOIN.

ЛИТЕРАТУРА:

1. Bakkum P., Skadron K. Accelerating SQL Database Operations on a GPU with CUDA // Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU'10), Pittsburgh, USA, March 14, 2010. ACM. P. 94-103.
2. Di Blas A., Kaldeway T. Datamonster: Why graphics processors will transform database processing. IEEE Spectrum, September, 2009.
3. Ding S., He J., Yan H., Suel T. Using graphics processors for high performance IR query processing. In WWW ’09: Proceedings of the 18th international conference on World wide web, New York, NY, USA, 2009. ACM. P. 421-430.
4. Fang W., Lau K., Lu M., et al. Parallel data mining on graphics processors: Technical report, Hong Kong University of Science and Technology, 2008.
5. Govindaraju N., Lloyd B., Wang W., et al. Fast computation of database operations using graphics processors. In SIGGRAPH ’05: ACM SIGGRAPH 2005 Courses, New York, NY, USA, 2005. ACM. P. 206.
6. Соколинский Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической структурой. // Программирование. 2001. № 6. С. 13-19.