УДК 004.65

ОБРАБОТКА ЗАПРОСОВ НА КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С МНОГОЯДЕРНЫМИ УСКОРИТЕЛЯМИ

П.С. Костенецкий

Работа посвящена вопросам моделирования процесса обработки запросов в мультипроцессорах параллельных систем баз данных. Предлагаются подходы для обработки запросов при помощи графических ускорителей, многоядерных сопроцессоров и центральных процессоров. На базе предложенных подходов реализован эмулятор параллельной СУБД. Приведены результаты вычислительных экспериментов и выполнен анализ эффективности предложенных подходов.

Ключевые слова: многоядерные ускорители, параллельная обработка запросов.

Введение

Важным классом приложений для многопроцессорных систем являются задачи, связанные с обработкой сверхбольших баз данных. Фактически единственным эффективным решением проблемы хранения и обработки сверхбольших баз данных является использование параллельных систем баз данных, обеспечивающих параллельную обработку запросов на многопроцессорных вычислительных системах. В области параллельных систем баз данных до сих пор остается ряд направлений, требующих дополнительных научных исследований [1]. Одно из них – дальнейшее развитие аппаратной архитектуры параллельных систем баз данных [19]. Разработка систем баз данных является одной из областей, где могут быть успешно применены многоядерные ускорители [3].

На сегодняшний день уже существуют работы, посвященные оптимизации процесса интеллектуального анализа данных [4, 17] и ускорению обработки запросов к базе данных с использованием графических ускорителей [2, 5]. Разработан алгоритм работы с индексом, учитывающий архитектурные особенности GPU [11]. Предложен метод ускорения операций над индексом, эффективно использующий большое число вычислительных ядер GPU [10]. Описаны алгоритмы сортировки [12]. В работе [9] описывается совместное использование GPU и CPU при обработке запросов, при котором запрос выполняется на GPU только в тех случаях, когда стоимость выполнения запроса на нем ниже, чем на центральном процессоре. Стоимость выполнения запроса вычисляется динамически на основе плана запроса. В работе [10] предлагаются алгоритмы группировки транзакций с целью их последующего выполнения на графическом ускорителе. Работа [6] предлагает использование GPU для оптимизации запросов. В работе [14] рассмотрена разработка прототипа СУБД, хранящей свои данные в памяти графического ускорителя. В некоторых работах проведена адаптация существующих СУБД для использования графических ускорителей и оценка эффективности такой адаптации. Так, в работе [8] поддержка вычислений с использованием GPU была интегрирована в СУБД Oracle 9, в работе [13] поддержка графических ускорителей была добавлена в СУБД WattDB, а в работе [5] использовалась СУБД SQLite.

Разработка параллельных систем баз данных [17], использующих многоядерные сопроцессоры и графические ускорители, осложнена наличием у данного оборудования ряда технических особенностей. В первую очередь, это ограничение скорости системной

шины, по которой данные передаются между центральным процессором и многоядерным ускорителем. Кроме того, ускорители имеют сравнительно небольшой объем оперативной памяти, что так же необходимо учитывать при составлении алгоритмов обработки баз данных. В связи с этим актуален вопрос оценки эффективности использования кластерных систем с многоядерными ускорителями и GPU для приложений баз данных. В свою очередь, для оценки эффективности подобных гибридных кластерных архитектур необходима разработка новых методов обработки запросов [16].

1. Методы обработки запросов

В ходе исследования были разработаны методы обработки наиболее распространенных типов запросов SELECT и JOIN. Методы были реализованы в виде программного эмулятора СУБД, позволяющего моделировать параллельную обработку запросов [18] к базе данных на многоядерном центральном процессоре и на графических ускорителях, поддерживающих технологию CUDA.

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 кортежа в набор, группировка значений, порядок вывода записей и др.). Запрос выполняется следующим образом: сначала извлекаются все 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.

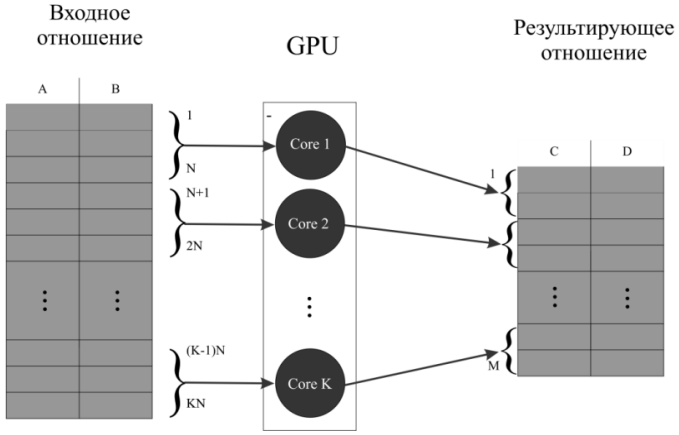

Рис. 1. Схема выполнения запроса SELECT на GPU

Алгоритм выполнения запроса SELECT на GPU состоит из следующих шагов:

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, при котором количество фрагментов равно количеству нитей CUDA;
5) выполнение выборки из отношения R на графическом ускорителе, при котором каждая нить обрабатывает один фрагмент, а запись результатов проводится в массив созданный ранее в памяти графического ускорителя;

6) копирование результата из памяти графического ускорителя в оперативную память.

Алгоритм выполнения запроса SELECT на многоядерном центральном процессоре состоит из следующих шагов:

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

Реализация запроса JOIN. В работе 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.

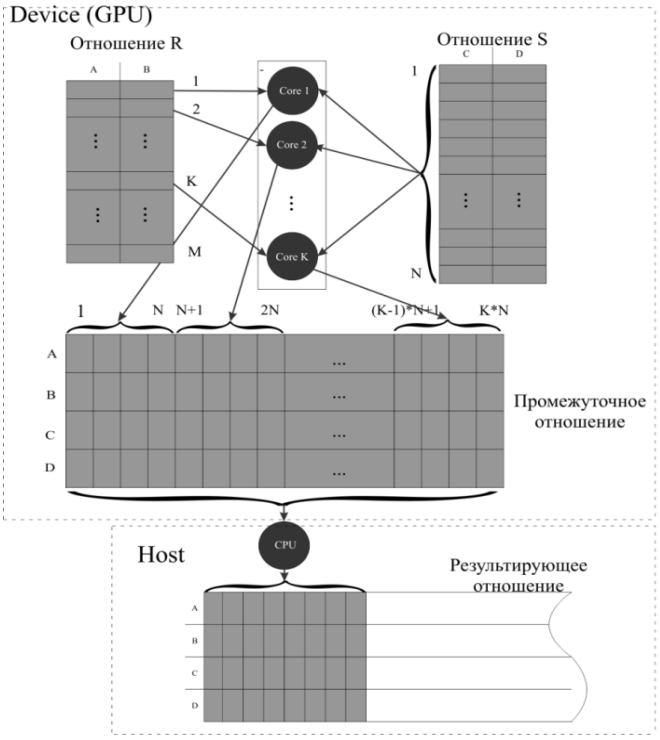

Рис. 2. Схема выполнения запроса JOIN

Алгоритм выполнения запроса INNER JOIN на графическом ускорителе состоит из следующих шагов:

1) генерация входных отношений R и S в оперативной памяти;
2) копирование отношений R и S из оперативной памяти в память графического ускорителя;
3) выделение памяти графического ускорителя под массив для результатов;
4) логическое разбиение отношения R на фрагменты, при котором количество кортежей во фрагменте равно количеству нитей CUDA;

5) выполнение соединения фрагмента отношения R и отношения S на графическом ускорителе, при котором каждая нить обрабатывает один кортеж R, а запись результатов проводится в массив, созданный ранее в памяти графического ускорителя;
6) копирование промежуточного результата из памяти графического ускорителя в оперативную память;
7) копирование промежуточного результата в результирующее отношение;
8) очистка массива для промежуточных результатов на GPU;
9) повторение шагов 5–9 для следующего фрагмента отношения R.

Алгоритм выполнения запроса JOIN на многоядерном центральном процессоре состоит из следующих шагов:

1) генерация входных отношений R и S;
2) выполнение соединения отношений R и S, при котором нити OpenMP в цикле обрабатывают кортежи и записывают результаты в результирующий динамический массив.

2. Вычислительные эксперименты

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

Для проведения вычислительных экспериментов была сгенерирована тестовая база данных, состоящая из двух отношений R и S. Размеры отношений для операции соединения: $|R|$=1 000, |S|=100 000 кортежей. Для операции выборки использовалась база данных состоящая из одного отношения содержащего 75 000 000 кортежей. Отношения состоят из двух атрибутов и содержат данные типа *Integer*.

Таблица 1

Характеристики используемого оборудования

Оборудование	Характеристики	
Процессор	Intel Core i7-2600 (4 ядра по 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
	Объем памяти	1024 Мб GDDR5 (4100 МГц)
	Разрядность памяти	192-bit
	Пропускная способность памяти	98,4 Гб/с

Исследование эффективности операции соединения для GPU. На рис. 3 приведена зависимость ускорения выполнения запроса INNER JOIN над тестовой базой данных на GPU при варьирующемся от 32 до 512 количестве нитей CUDA. Следует отметить, что используемый графический ускоритель имеет 192 физических ядра.

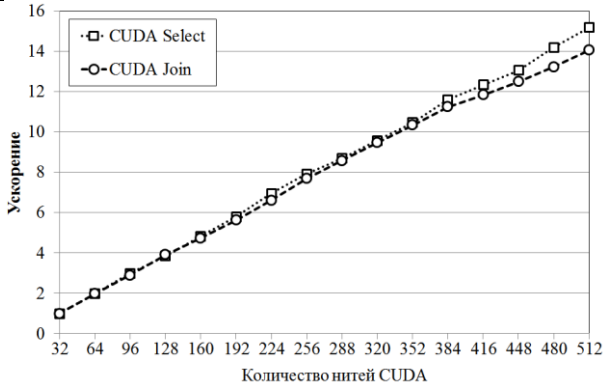

Рис. 3. Ускорение обработки запросов на GPU

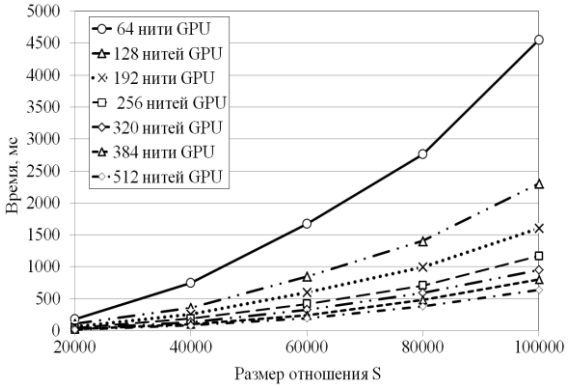

Рис. 4. Время выполнения запроса INNER JOIN на GPU

На рис. 4 приведено время выполнения запроса INNER JOIN на графическом ускорителе при варьирующемся количестве кортежей.

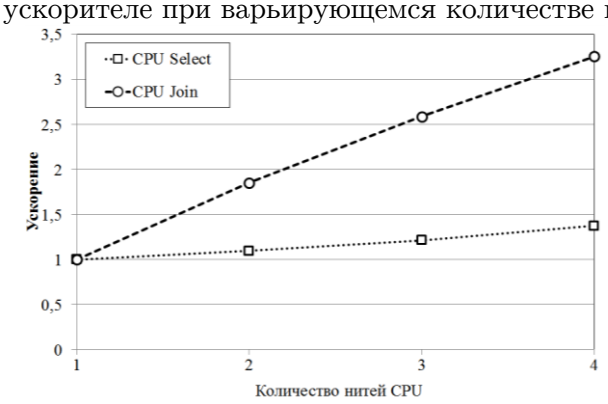

Рис. 5. Ускорение обработки запросов на CPU

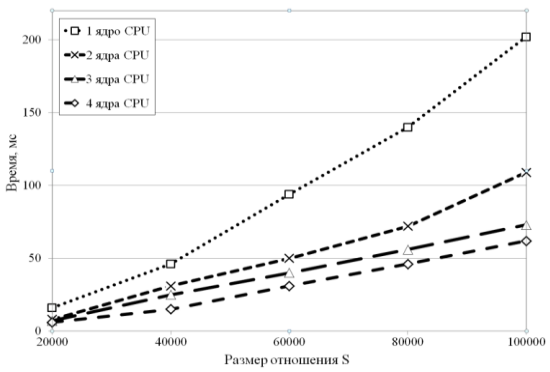

Рис. 6. Время выполнения запроса INNER JOIN на CPU

Исследование эффективности соединения для CPU. Для сравнения производительности различных аппаратных архитектур на рис. 5 и 6 приводятся графики ускорения и времени обработки запроса INNER JOIN на центральном процессоре при варьирующемся количестве нитей OpenMP. При реализации алгоритма JOIN на CPU использование количества нитей, превосходящего количество физических ядер, приводит к значительному снижению производительности, в отличие от реализации на CUDA. 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 на GPU при варьирующемся количестве нитей CUDA (см. рис. 3), параллельная реализация операции SELECT на CUDA дает ускорение в 11,5 раза при использовании 512 нитей по сравнению с обработкой на 32 нитях.

Исследование эффективности операции выборки для CPU. Из графика ускорения обработки тестовой базы данных на CPU при варьирующемся количестве нитей OpenMP, представленного на рис. , видно, что для операции SELECT увеличение количества ядер приводит к небольшому приросту производительности. Например, 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.

Если сравнивать производительность CPU и GPU, то для запроса SELECT использование GPU является более эффективным. При помощи технологии CUDA на бюджетном графическом ускорителе GeForce GTX 550 Ti достигнута эффективность ускорения работы запроса SELECT в 83 раза при обработке на 512 нитях GPU по сравнению с обработкой на четырех ядрах CPU Intel Core i7.

Заключение

В результате исследования определено, что с использованием многоядерных графических ускорителей можно добиться многократного ускорения обработки запроса SELECT или вычисления агрегатных функций. По сравнению с обработкой на 4 ядрах центрального процессора удалось добиться ускорения в 83 раза при обработке на 512 нитях GPU. В то же время значительно ускорить алгоритм JOIN с использованием GPU не удается.

Исходя из результатов экспериментов, можно сделать вывод, что эффективным будет следующий комбинированный алгоритм работы системы баз данных. Обработка запроса JOIN выполняется на ядрах центрального процессора. В это же время на графическом ускорителе, установленном в вычислительном узле, выполняется обработка запросов SELECT или вычисление агрегатных функций, таких как MIN, MAX, SUM, AVG или COUNT. SELECT является наиболее распространенным запросом SQL, поэтому использование такой схемы работы может позволить значительно снизить нагрузку на центральные процессоры. Для реализации данного подхода может быть эффективным использование нескольких многоядерных ускорителей в рамках одного вычислительного узла кластера. Для реализации подобного комбинированного подхода к обработке запросов в сверхбольших базах данных потребуются новые аппаратные архитектуры систем баз данных, содержащие ускорители GPU либо MIC. Сборка прототипов подобных высокопроизводительных кластеров с гибридными узлами является чрезвычайно дорогостоящей. В связи с этим для выбора оптимальной конфигурации системы баз данных предварительно необходимо выполнять моделирование возможных аппаратных архитектур при помощи математических моделей [15].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-31082.

Литература

1. Agrawal R., Ailamaki A., Bernstein P.A. et al. The Claremont Report on Database Research // Communications of the ACM, 2009. – Vol. 52, No. 6. – P. 56–65.

2. Bakkum P., Skadron K. Accelerating SQL Database Operations on a GPU with CUDA // The 3rd Workshop on General-Purpose Computation on Graphics Processing Units, Pittsburgh, USA, March 14, 2010, Proceedings. ACM, 2010. – P. 94–103.
3. Blas A.D., Kaldewey T. Data Monster // IEEE spectrum, 2009. – Vol. 46, No. 9.
4. Ding S., He J., Yan H., Suel T. Using Graphics Processors for High Performance IR Query Processing // The 18th international conference on World Wide Web, New York, USA, April 20– 24, 2009, Proceedings. ACM, 2009. – P. 421–430.
5. Govindaraju N., Lloyd B., Wang W., et al. Fast Computation of Database Operations Using Graphics Processors // ACM SIGGRAPH 2005 Courses, New York, USA. ACM, 2005. – P. 206.
6. Heimel M., Markl V. A First Step Towards GPU-assisted Query Optimization // The Third International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures, Istanbul, Turkey, August 27, 2012. – P. 1–12.
7. Bakkum P., Skadron K. Accelerating SQL Database Operations on a GPU with CUDA // 3rd Workshop on General Purpose Computation on Graphics Processing Units, New York, USA, March 14, 2010, Proceedings. ACM, 2010. – P. 94–103
8. Bandi N., Sun C., Agrawal D., Abbadi A.E. Hardware Acceleration in Commercial Databases: a Case Study of Spatial Operations. The 30th International Conference on Very Large Data Bases, August 31 – September 3, 2004, – Vol. 30, Proceedings. VLDB Endowment, 2004. – P. 1021–1032.
9. He B., Lu M., Yang K., Fang R, et.al. Relational query coprocessing on graphics processors. ACM Trans. Database Syst., – Vol. 34(4), ACM, 2009. – P. 21:1–21:39
10. He B., Xu Y.J. Highthroughput transaction executionson graphics processors // VLDB Endowment, Seattle, Washington, USA, August 29 – September 3, 2011, Proceedings. – Vol. 4, No. 5. VLDB Endowment, 2011. – P. 314–325.
11. Kim C., Chhugani J., Satish N. FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs // ACM SIGMOD International Conference on Management of data, Indianapolis, USA, June 6–10, Proceedings. ACM, 2010. – P. 339–350.
12. Satish N., Kim C., Chhugani J, et. al. Fast Sort on CPUs and GPUs: a Case for Bandwidth Oblivious SIMD Sort. The 2010 ACM SIGMOD International Conference on Management of data, New York, USA, 2010, Proceedings. ACM. 2010. – P. 351–362
13. Vitor U.R. A GPU Operations Framework for Wattdb. Technical report. – Germany, Kaiserslautern: University of Kaiserslautern, 2012.
14. Hansen C.E. Christiansen M. CUDA DBMS. Technical report. – Denmark, Copenhagen: Aalborg University, 2009.
15. Костенецкий П.С., Лепихов А.В., Соколинский Л.Б. Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. – 2007. № 5. – С. 112–125.
16. Лепихов А.В., Соколинский Л.Б. Обработка запросов в СУБД для кластерных систем // Программирование. 2010. – № 4. – С. 25–39.
17. Пан К.С., Цымблер М.Л. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". 2012. – Вып. 12. – № 18(277).– С. 112–120.

18. Соколинский Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической структурой // Программирование. 2001. – № 6. – С. 13–19.
19. Соколинский Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. 2004. – № 6. – С. 49–63.

Костенецкий Павел Сергеевич, к.ф.-м.н., директор суперкомпьютерного центра, Южно-Уральский государственный университет, Kostenetskiy@gmail.com

Query Processing on Cluster Based Sysems with Multicore Acelerators

P.S. Kostenetskiy, South Ural State University (Chelyabinsk, Russian Federation),

The paper is devoted to the problem of modeling query execution process in multiprocessors of parallel database systems. Original approaches to the query execution process on GPU, MIC and multicore CPU are presented. Based on this approach, a simulator of parallel DBMS is developed. Results of computational experiments are presented, and analysis of efficiency of the proposed approaches is performed.

Keywords: parallel query processing, database multiprocessor model.

References

1. Agrawal R., Ailamaki A., Bernstein P.A. et al. The Claremont Report on Database Research // Communications of the ACM, 2009. – Vol. 52, No. 6. – P. 56–65.
2. Bakkum P., Skadron K. Accelerating SQL Database Operations on a GPU with CUDA // The 3rd Workshop on General-Purpose Computation on Graphics Processing Units, Pittsburgh, USA, March 14, 2010, Proceedings. ACM, 2010. – P. 94–103.
3. Blas A.D., Kaldewey T. Data Monster // IEEE spectrum, 2009. – Vol. 46, No. 9.
4. Ding S., He J., Yan H., Suel T. Using Graphics Processors for High Performance IR Query Processing // The 18th international conference on World Wide Web, New York, USA, April 20–24, 2009, Proceedings. ACM, 2009. – P. 421–430.
5. Govindaraju N., Lloyd B., Wang W., et al. Fast Computation of Database Operations Using Graphics Processors // ACM SIGGRAPH 2005 Courses, New York, USA. ACM, 2005. – P. 206.
6. Heimel M., Markl V. A First Step Towards GPU-assisted Query Optimization // The Third International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures, Istanbul, Turkey, August 27, 2012. – P. 1–12
7. Bakkum P. and Skadron K. Accelerating SQL Database Operations on a GPU with CUDA // 3rd Workshop on General Purpose Computation on Graphics Processing Units, New York, USA, March 14, 2010, Proceedings. ACM, 2010. – P. 94–103
8. Bandi N., Sun C., Agrawal D., Abbadi A.E. Hardware Acceleration in Commercial Databases: a Case Study of Spatial Operations. The 30th International Conference on Very Large Data Bases, August 31 – September 3, 2004, – Vol. 30, Proceedings. VLDB Endowment, 2004. – P. 1021–1032.
9. He B., Lu M., Yang K., Fang R, et.al. Relational query coprocessing on graphics processors. ACM Trans. Database Syst., – Vol. 34(4), ACM, 2009. – P. 21:1–21:39

10. He B., Xu Y.J. Highthroughput transaction executionson graphics processors // VLDB Endowment, Seattle, Washington, USA, August 29 – September 3, 2011, Proceedings. – Vol. 4, No. 5. VLDB Endowment, 2011. – P. 314–325.
11. Kim C., Chhugani J., Satish N. FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs // ACM SIGMOD International Conference on Management of data, Indianapolis, USA, June 6–10, Proceedings. ACM, 2010. – P. 339–350.
12. Satish N., Kim C., Chhugani J, et. al. Fast Sort on CPUs and GPUs: a Case for Bandwidth Oblivious SIMD Sort. The 2010 ACM SIGMOD International Conference on Management of data, New York, USA, 2010, Proceedings. ACM. 2010. – P. 351–362
13. Vitor U.R. A GPU Operations Framework for Wattdb. Technical report. – Germany, Kaiserslautern: University of Kaiserslautern, 2012.
14. Hansen C.E. Christiansen M. CUDA DBMS. Technical report. – Denmark, Copenhagen: Aalborg University, 2009.
15. Kostenetskii P.S., Lepikhov A.V., Sokolinskii L.B. Technologies of parallel database systems for hierarchical multiprocessor environments // Automation and Remote Control. 2007. – Vol. 68, No. 5. – P. 847-859.
16. Lepikhov A.V., Sokolinsky L.B. Query Processing in a DBMS for Cluster Systems // Programming and Computer Software. 2010. – Vol. 36. No. 4. – P. 205–215.
17. Pan C.S., Zymbler M.L. Razrabotka parallelnoj SUBD na osnove posledovatelnoj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS]. Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software]. 2012. No. 18(277). Vol. 12. P. 112–120.
18. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture // Programming and Computer Software. 2001. – Vol. 27. No. 6. – P. 297–308.
19. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems // Programming and Computer Software. 2004. – Vol. 30. № 6. – P. 337–346.

Поступила в редакцию 5 ноября 2012 г.