УДК 004.021
DOI: 10.12737/20288

А.А. Колпаков, Ю.А. Кропотов

АЛГОРИТМ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ГЕТЕРОГЕННЫХ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Представлен разработанный алгоритм повышения производительности гетерогенных многопроцессорных вычислительных систем. Предложено использование графического процессора в качестве дополнительного вычислителя.

Ключевые слова: GPGPU, PRAM-модель, параллельные вычисления, гетерогенные вычислительные системы, графические процессоры.

A.A. Kolpakov, Yu.A. Kropotov

ALGORITHM FOR EFFICIENCY INCREASE OF HETEROGENEOUS MULTI-PROCESSOR COMPUTER SYSTEMS

In the paper there is considered an algorithm developed on the basis of modified PRAM-model for efficiency increase of parallel computations on specialized computer modules.

By means of the efficiency assessment method there were carried out comparative experimental investigations of the algorithm developed. The assessment results of the algorithm for the parallel computation efficiency increase on special computer modules show efficiency increase not less than 2-4 times depending on the number of flows under investigation.

Key words: GPGPU, PRAM-model, parallel computations, heterogeneous computer systems, graphics processors.

Введение

Известно, что повышение эффективности вычислительных компьютерных систем осуществляется в зависимости от организации процесса решения задач [1; 2]. В общем случае задачи представляются параллельными программами и описываются рядом параметров, в числе которых: количество ветвей, ранг необходимой подсистемы, время решения и т.п. Режим функционирования высокопроизводительных вычислительных систем формируется мультипрограммным методом. В некоторых вычислительных компьютерных системах осуществляется частичное применение вычислительных модулей, что в недостаточной степени обеспечивает повышение производительности вычислений [3].

В связи с этим возникает задача разработки методов повышения производительности компьютерных систем на основе модели архитектуры с использованием дополнительных вычислительных модулей или однородных модулей на графических процессорах. Основной задачей повышения производительности такой вычислительной системы является решение проблемы принятия решений о переносе операций вычислений на специализированные вычислительные модули и кэшировании данных, что требует исследований и разработки соответствующих алгоритмов [4].

Архитектура гетерогенных многопроцессорных вычислительных систем

Для рассмотрения особенностей обобщенной архитектуры специализированных вычислительных модулей и их взаимодействия с центральным процессором была разработана и исследована структурная схема архитектуры гетерогенной многопроцессорной вычислительной системы (рис. 1). Базовыми структурными элементами специализированных вычислительных модулей являются спецпамять

(SpRAM), в которой отдельно можно выделить память констант и глобальную память, и множество мультипроцессоров. Чтобы обработать данные на специализированных вычислительных модулях, необходимо передать их из оперативной памяти компьютера в SpRAM в соответствии со структурной схемой архитектуры гетерогенной системы на рис. 1.

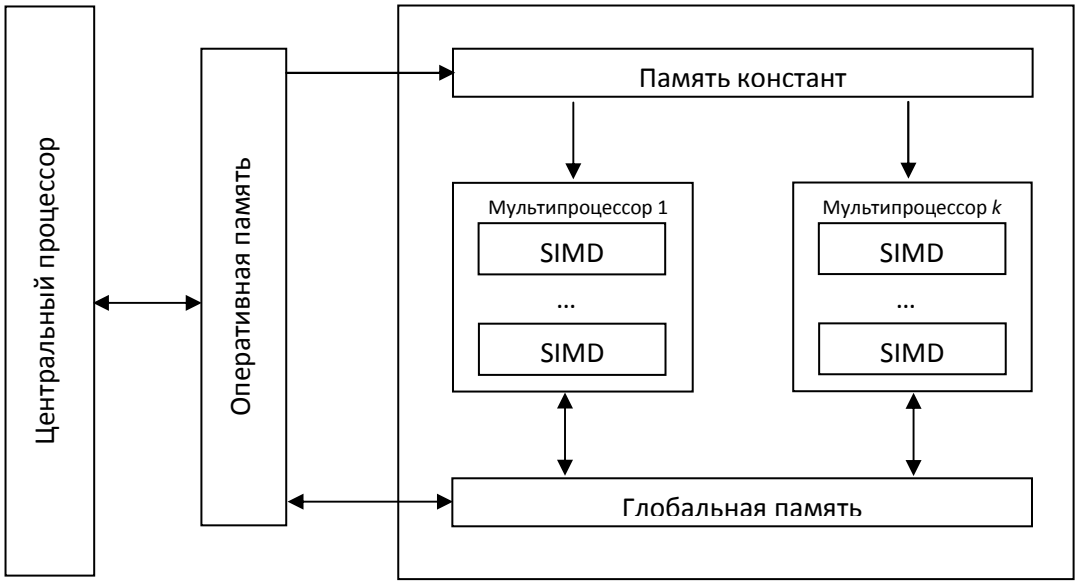


Рис. 1. Структурная схема архитектуры гетерогенной многопроцессорной вычислительной системы

Как видно из структурной схемы, связку «центральный процессор – графический процессор» можно отнести к модели с общей памятью. Основной моделью с общей памятью является модель PRAM (parallel random-access machine) – машина с параллельным произвольным доступом. Она является абстрактной идеализированной моделью параллельной синхронной машины с разделяемой общей памятью, которая использует следующие допущения:

−количество процессоров (q) в машине не ограничено;

−каждый процессор имеет равнозначный доступ к любой ячейке общей памяти, размер которой не ограничен;

−отсутствует конкуренция по ресурсам;

−процессоры работают в режиме MIMD, но в частном случае может использоваться режим SIMD.

Все процессоры исполняют инструкции синхронно, причем выполнение любой инструкции занимает ровно 1 такт, называемый шагом PRAM-машины.

Чтобы оценить время выполнения алгоритма для N элементов входных данных на PRAM-машине с p потоками, было получено выражение

$$T(N,p) = O\left(\frac{W(N)}{p} + S(N)\right), \qquad (1)$$

где O – верхняя асимптотическая оценка трудоёмкости алгоритма; N – количество входных данных алгоритма; $S(N)$ – шаговая сложность алгоритма; $W(N) = \sum_{i=1}^{S(n)} W_i(N)$ – рабочая сложность параллельного алгоритма, ($W_i(N)$ – количество параллельных операций на шаге i).

Формула (1) дает верхнюю асимптотическую оценку времени выполнения алгоритма с шаговой сложностью $S(N)$ и рабочей сложностью $W(N)$.

Следует отметить, что PRAM-модель может быть применена к многопроцессорной системе (рис. 1) с учётом следующих уточнений и дополнений:

1. Все процессоры могут одновременно считывать данные из разделяемой памяти, но запись должна быть монопольной, так как порядок изменения ячейки

разделяемой памяти при обращении на запись из нескольких скалярных процессоров не определён (PRAM – CREW (concurrent read, exclusive write)).

2. Количество скалярных процессоров в графическом мультипроцессоре ограничено сверху (q_{max} процессоров). Для выполнения большего числа потоков используется система горизонтального параллелизма, аналогичная горизонтальной структуре в модели BSP (генерируется расписание последовательного исполнения потоков, разбитых на пучки по q_{warp} скалярных процессоров).

3. Размер разделяемой памяти мультипроцессора ограничен – M_s байт.

4. Все скалярные процессоры работают по принципу SIMD с одинаковой скоростью S_{GPU} элементарных операций в секунду.

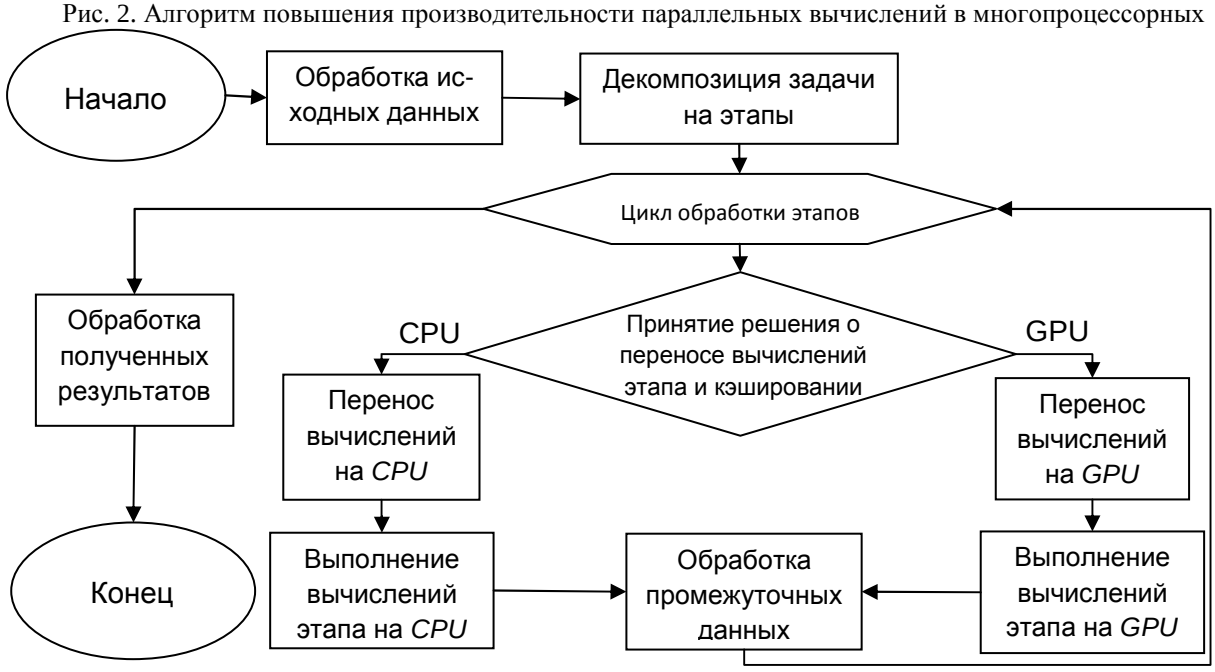
5. Должна быть дополнительная операция – обращение к оперативной памяти SpRAM специализированного вычислительного модуля на чтение или запись. Задержка при обращении K определяется количеством элементарных операций, требуемых при обращении к одному числу одинарной точности в глобальной памяти специализированного вычислительного модуля.

Таким образом, PRAM-модель с перечисленными уточнениями и дополнениями допускает применение графических процессоров в качестве специализированных вычислительных модулей для общих вычислений.

Общий алгоритм оптимизации параллельных вычислений в многопроцессорных вычислительных системах с гетерогенной архитектурой

Для организации параллельных вычислений в многопроцессорных вычислительных системах с гетерогенной архитектурой CPU–SCM был разработан общий алгоритм оптимизации (рис. 2). В качестве специализированного вычислительного модуля используется графический процессор GPU.

Рис. 2. Алгоритм повышения производительности параллельных вычислений в многопроцессорных



вычислительных системах с гетерогенной архитектурой

Основным этапом разработанного алгоритма является принятие решения о переносе вычислений этапа на графический процессор. Для осуществления сравнения производительности этапа алгоритма на различных вычислительных устройствах и последующего принятия решения о переносе вычислений используется модифицированная PRAM-модель.

Модифицированная PRAM-модель

В соответствии с моделью специализированного мультипроцессора, которая является общей для всех графических процессоров, сформирован абстрактный вычислительный мультипроцессор. Для абстрактного вычислительного мультипроцессора имеем множество параметров, учитывающих основные характеристики реальных специализированных мультипроцессоров, $\{q_{max}, q_{warp}, M_s, S_{GPU}, K\}$. Для разработки параллельного алгоритма под предложенную модель можно воспользоваться методом создания расписания распределения потоков вычислений, который применяется в базовой PRAM-модели, учитывая изложенные выше уточнения и дополнения. В этом случае формулу (1) для верхней оценки времени выполнения алгоритма на PRAM-машине следует скорректировать. PRAM-модель теперь должна быть представлена в виде одного абстрактного вычислительного мультипроцессора, на котором все скалярные процессоры работают пучками по принципу горизонтального параллелизма. Выражение для вычисления верхней оценки временной сложности алгоритма принимает вид

$$T_C(N,p) = O\left(\frac{W(N)}{p} ceil\left(\frac{p}{q_{warp}}\right) + S(N)\right), \quad (2)$$

где *ceil* – функция округления дроби до ближайшего большего целого числа; p – число потоков алгоритма, предназначенных для обработки N элементов данных, $p < q_{max}$.

Основным объектом исследования является учет операций обращения к глобальной памяти графического процессора. Необходимо ввести дополнительный параметр алгоритма – сложность обращения к глобальной памяти $R(N)$, т.е. суммарное количество обращений на чтение и запись из глобальной памяти графического процессора, требуемое для обработки N элементов данных. Данный вид операций должен присутствовать в любом параллельном алгоритме для графических процессоров, который обрабатывает входные данные. Так как процессоры работают в режиме SIMD и выполняют команды последовательно пучками по принципу горизонтального параллелизма, то формула для верхней оценки времени выполнения параллельного алгоритма на одном абстрактном вычислительном мультипроцессоре принимает вид

$$T_C^{GPU}(N,p) = O\left(\frac{W(N)+R(N)}{p} ceil\left(\frac{p}{q_{warp}}\right) + S(N)\right). \quad (3)$$

Из выражения (3) следует, что более высокая производительность будет у того алгоритма, который будет иметь меньшее количество обращений к SpRAM. Тогда выражение для определения верхней оценки времени выполнения алгоритма на одном абстрактном вычислительном мультипроцессоре имеет вид

$$T_M(N,p) = \frac{W_M(N) + R_M(N)K}{S_{GPU}\, p} ceil\left(\frac{p}{q_{warp}}\right),$$

где $W_M(N)$ – количество элементарных операций одного процессора абстрактного вычислительного мультипроцессора в PRAM; $R_M(N)$ – количество обращений к SpRAM из одного процессора абстрактного вычислительного мультипроцессора в PRAM.

Для учета передачи данных между оперативной памятью и памятью SpRAM следует ввести ещё два дополнительных параметра: суммарное количество входных данных этапа алгоритма в байтах $N`_{HD}$ и суммарное количество выходных данных этапа алгоритма в байтах $N`_{DH}$. Тогда выражение для вычисления общего времени работы этапа алгоритма принимает вид

$$T'_{GPU}(N) = \frac{N'_{HD}(N)}{S_{HD}} + T'_G(N) + \frac{N'_{DH}(N)}{S_{DH}},$$

где S_{HD} и S_{DH} – константы скорости передачи данных между RAM и SpRAM (байт/с).

Полученная модель показывает, что для анализа и сравнения параллельных алгоритмов необходимо использовать следующие параметры алгоритма:

1. Суммарная шаговая сложность

$$S(N) = \sum_{i=1}^{B(N)} S'_i(N). \quad (4)$$

2. Суммарная рабочая сложность

$$W(N) = \sum_{i=1}^{B(N)} W_i'(N) \quad . \quad (5)$$

3. Суммарная сложность обращения к глобальной памяти специализированного вычислительного модуля

$$R(N) = \sum_{i=1}^{B(N)} R_i'(N) \quad . \quad (6)$$

4. Суммарный объём данных, передаваемых между оперативной памятью вычислительной компьютерной системы и глобальной памятью специализированного вычислительного модуля,

$$N_{HD}(N) = \sum_{i=1}^{B(N)} N'_{i\,HD}(N)$$

$$N_{DH}(N) = \sum_{i=1}^{B(N)} N'_{i\,DH}(N) \quad . \quad (7)$$

С учетом выражений (4-7) верхняя оценка времени работы алгоритма на графическом процессоре в среде CPU-GPU вычисляется следующим образом:

$$T_{GPU}(N) = \frac{N_{HD}(N)}{S_{HD}} + \sum_{i=1}^{B(N)} T'_{i\,G}(N) + \frac{N_{DH}(N)}{S_{DH}} \quad . \quad (8)$$

При принятии решения о переносе вычислений на GPU предварительно оценивается время выполнения алгоритма на CPU и GPU в соответствии с выражениями (2) и (8). После этого осуществляется сравнение полученных временных показателей, по результату которого принимается решение о переносе вычислений.

Экспериментальное исследование разработанного алгоритма

В качестве тестовой задачи использовалась задача нахождения нулевых битовых векторов, решаемая с применением генетических алгоритмов [5]. При решении указанной задачи основное время работы занимают параллельные вычисления значений функции приспособленности различных особей, операции скрещивания и мутации. Используемый алгоритм ее решения имеет свойства, характерные для многих генетических алгоритмов:

1. Представление особи в виде битовой строки.

2. Малое число логических операций при вычислении функции приспособленности, выполнении мутации и скрещивания.

3. Последовательный доступ к памяти.

Данные свойства позволяют эффективно применять вычисления на графическом процессоре.

Для проведения экспериментальной оценки эффективности работы алгоритма оптимизации использовалась тестовая компьютерная система следующей конфигурации: центральный процессор Intel Core 2 Quad Q9400 (2.66GHz), ОЗУ 8GB, графическая карта Nvidia GeForce GTX560 2Gb (336 потоков), операционная система Windows 7 x64**,** компилятор MS Visual Studio 2008 (в режиме release).

При исследовании производительности изменялось количество 32-битных целых чисел в массиве (*M*) и число параллельных потоков (*N*).

Определялось среднее время *t*, потраченное на получение нового поколения для различного количества 32-битных целых чисел в массиве и числа параллельных потоков. Исследования проводились с использованием технологий OpenCL и NVIDIA CUDA.

Результаты экспериментальных исследований приведены в таблице.

Таблица

Время генерации (мс) многопроцессорной системой одного поколения (N =10)

Процессор	Количество особей в поколении				
	128	1024	10240	102400	1024000
CPU - Q9400	0,38	0,56	2,5	22,2	416,2
CUDA GPU - GTX460	0,08	0,14	1,03	13	237,4

Как видно из результатов экспериментального исследования, применение разработанного алгоритма оптимизации дает рост производительности относительно центрального процессора: в случае использования NVIDIA CUDA время обработки сокращается с 0,38 мс до 0,08 для 128 потоков и с 416,2 до 237,4 мс для 102400 потоков.

Заключение

Таким образом, на основе модифицированной PRAM-модели разработан алгоритм повышения производительности параллельных вычислений на специализированных вычислительных модулях, который включает в себя алгоритм принятия решения о переносе вычислений на графический процессор.

Методом оценивания производительности были осуществлены сравнительные экспериментальные исследования разработанного алгоритма. Результаты оценивания алгоритма показывают повышение производительности не менее чем в 2-4 раза в зависимости от числа исследуемых потоков.

СПИСОК ЛИТЕРАТУРЫ

1. Воеводин, В.В. Параллельные вычисления: новые концепции в науке и образовании / В.В. Воеводин, Вл.В. Воеводин // Современные проблемы вычислительной математики и математического моделирования. В 2 т. Т.1. Вычислительная математика. – Наука, 2005. – С.2-15.
2. Graham, R.L. Bounds on Multiprocessing Timing Anomalies / R.L. Graham // SIAM Journal on Applied Mathematics. – 1969. - Vol. 17. - № 2. - P. 416-429.
3. Колпаков, А.А. Аспекты оценки увеличения производительности вычислений при распараллеливании процессоров вычислительных систем / А.А. Колпаков, Ю.А. Кропотов // Методы и устройства передачи и обработки информации. - 2011. – №1(13). – С.124-127.
4. Колпаков, А.А. Теоретическая оценка роста производительности вычислительной системы при использовании нескольких вычислительных устройств / А.А. Колпаков // В мире научных открытий. - 2012. – №1. – С. 206-209.
5. Колпаков, А.А. Оптимизация генетических алгоритмов при использовании вычислений на графических процессорах на примере задачи нулевых битовых векторов / А.А. Колпаков // Информационные системы и технологии. - 2013. – №2(76). – С. 22-28.

1. Voevodin, V.V. Parallel computations: new concepts in science and education / V.V. Voevodin, Vl.V. Voevodin // Modern Problems in Calculus Mathematics and Mathematic Modeling in 2 Vol. / Vol.1. Calculus Mathematics. – Science, 2005. – pp. 2-15.
2. Graham, R.L. Bounds on Multiprocessing Timing Anomalies / R.L. Graham // SIAM Journal on Applied Mathematics. – 1969. - Vol. 17, No. - № 2. - P. 416-429.
3. Kolpakov, A.A. Aspects of assessment of computation efficiency increase at paralleling processors of computer systems / A.A. Kolpakov, Yu.A. Kropotov // Methods and Devices for Information Transfer and Processing. - 2011. – №1(13). – pp.124-127.
4. Kolpakov, A.A. Theoretical assessment of computer system efficiency increase using some computer devices / A.A. Kolpakov // In the World of Scientific Discoveries. - 2012. – №1. – pp. 206-209.
5. Kolpakov, A.A. Genetic algorithms optimization using computations on graphic processors by example of problem of null bit vectors / A.A. Kolpakov // Information Systems and Technologies. - 2013. – №2(76). – pp. 22-28.

Статья поступила в редколлегию 27.01.2016.
Рецензент: д.т.н., профессор Брянского государственного технического университета Киричек А.В.

Сведения об авторах:

Колпаков Александр Анатольевич, ст. преподаватель Муромского института (филиала) Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых, e-mail: desT.087@gmail.com.

Кропотов Юрий Анатольевич, д.т.н., профессор Муромского института (филиала) Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых, e-mail: kaf-eivt@yandex.ru.

Kolpakov Alexander Anatolievich, Senior lecturer of Murom Institute (Branch), Stoletovs State University of Vladimir, e-mail: desT.087@gmail.com.

Кропотов Юрий Анатольевич, D.Eng., Prof. of Murom Institute (Branch), Stoletovs State University of Vladimir, e-mail: kaf-eivt@yandex.ru.