УДК 519.6

А. А. Шарамет

АЛГОРИТМ И МОДЕЛЬ ХРАНЕНИЯ ДАННЫХ ПРИ РЕШЕНИИ ЗАДАЧИ ВЗАИМОДЕЙСТВИЯ СПУТНИКА И ПЛАЗМЫ МЕТОДОМ МОЛЕКУЛЯРНОЙ ДИНАМИКИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

Рассмотрен параллельный алгоритм математической модели взаимодействия заряженного малого спутника и тепловой космической плазмы. Основной проблемой при реализации алгоритма является постоянное поддержание всего объема данных для каждого вычислителя в актуальном состоянии, а также синхронизация и минимизация обменов. Выбор графических ускорителей для расчета позволяет сфокусировать максимальное количество вычислителей на одном узле. Проблемы

Вестник Балтийского федерального университета им. И. Канта. 2015. Вып. 4. С. 114 – 120.

синхронизации решаются на уровне потоков процессора. Полученный алгоритм сложен в реализации, но наиболее эффективно использует вычислительные ресурсы для решения задачи, легко масштабируется для гибридных вычислений и имеет потенциал роста при использовании MPI+CUDA+CPU threads гибридной модели программирования.

This article considers a parallel algorithm of the mathematical model of interaction between a charged small satellite and thermal space plasma. The main problem of the algorithm implementation is to maintain the total amount of current data for each calculator and to synchronize and minimize exchanges. The range of graphics accelerators chosen for calculation makes it possible to focus the maximum number of calculators on a single node. Synchronization problems are solved at the level of processor threads. The resulting algorithm is difficult to implement; however, it shows the most efficient use of computing resources in solving the problem, can be easily scaled for hybrid computing, and has great potential when using the MPI + CUDA + CPU threads of a hybrid programming model.

Ключевые слова: математическое моделирование, метод молекулярной динамики, параллельное программирование.

Key words: mathematical modeling, molecular dynamics method, parallel programming.

Введение

Проблема взаимодействия электрически заряженного спутника и тепловой плазмы многократно рассматривалась в научной литературе, начиная с классической работы [1]. Хорошо известно, что потенциал спутника существенно искажает измерения потоков тепловых ионов [2—5]. Для уменьшения влияния этих эффектов применяются эквипотенциализация поверхности спутника [6], устройства для активного регулирования потенциала [7; 8]. Для математического моделирования широко использовался гидродинамический подход и метод крупных частиц [9—11]. С активным ростом мощности суперкомпьютеров в последнее время активно используются методы молекулярной динамики [12—15]. Рассмотрим широко известную задачу взаимодействия спутника и плазмы. Построение модели подробно рассмотрено в статье [12], где приведены алгоритмы для моделирования и способ хранения данных в памяти для решения задачи на двух и более графических сопроцессорах одного узла.

Один графический сопроцессор

Предположим, что количество частиц 10^4. Графический сопроцессор (GPU) может порождать тысячи потоков (нитей). Эффективное количество обрабатываемых за единицу времени частиц можно рассчитать по формуле $Количество = Max \cdot Count$, где Max — количество нитей одного мультипроцессора, $Count$ — количество мультипроцессоров.

Для Tesla s2050 количество расчетных модулей равно 21504. Таким образом, в выбранном случае вычислительные мощности превышают

или равны объему обрабатываемой информации. Поставим во взаимно-однозначное соответствие одну частицу одной нити. Под этим будем понимать следующее: нить производит все необходимые операции для соответствующей ей частицы в течение одного шага по времени.

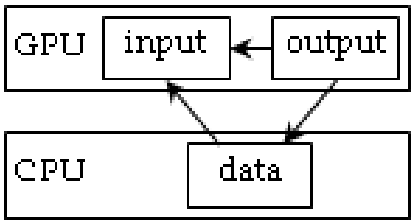

Рис. 1. Данные памяти

Рассмотрим хранение наших данных в памяти. Массив входных данных (data) загружается в оперативную память. Создаются два массива данных (input, output), равных по размеру массиву data и хранящихся в оперативной памяти графического процессора (DRAM). Производится копирование данных из data в input (рис. 1).

Каждая нить получает данные из input, производит преобразования по формулам, описанным в [13], и результат сохраняет в ячейке output. Если мы рассматриваем один шаг, то данные из output копируются в data и далее сохраняются для интерпретации. Но алгоритм состоит из множества повторяющихся шагов, где результат предыдущего шага используется как входные данные текущего; соответственно, в большинстве случаев данные из output будут скопированы в input.

Очевидно, что input и output исполняют функцию буфера, поэтому при организации прямой адресации в CPU задержки на передачу данных будут рассчитываться как сумма задержек внутри графической платы и PCI Express шины. Каждая нить для пересчета положения одной частицы обращается ко всем данным из input, следовательно, при прямой адресации объем передаваемого трафика возрастает многократно, что приводит к существенному замедлению алгоритма.

Разберем способы взаимодействия нитей внутри графической карты. В состав GPU входит несколько мультипроцессоров (SM), которые способны поддерживать работу от нескольких сотен до тысяч потоков. Каждый SM оснащен встроенной высокоскоростной памятью (shared memory), общей для всех нитей этого SM. Возьмем две соседних нити с номерами 0 и 1, каждая загружает в регистры уникальную для нее информацию (пересчитываемую частицу), далее обходит весь массив данных, считывая по одной частице, и использует их для расчета частичных сумм компонент вектора электрической напряженности. Предположим, что 0-й поток вытащил *n*-ю частицу и сделал копию в shared memory. Тогда 1-й поток обработает эту частицу намного быстрее в связи с тем, что скорость доступа в shared memory многократно больше, чем в DRAM. На этом принципе и построена оптимизация алгоритма.

Разобьем весь массив входных данных на блоки (по количеству исполняемых в текущий момент времени нитей на SM). С точки зрения нити каждый блок загружается за одну операцию, так как количество загружаемых элементов равно количеству нитей и загрузка происходит параллельно. После того как все данные блока загружены, нити приступают к частичному пересчету, а далее таким же способом загружа-

ются остальные блоки. Очевидно, что это практически полностью позволило заменить операции чтения из DRAM на операции чтения из shared memory и значительно повысило производительность (рис. 2).

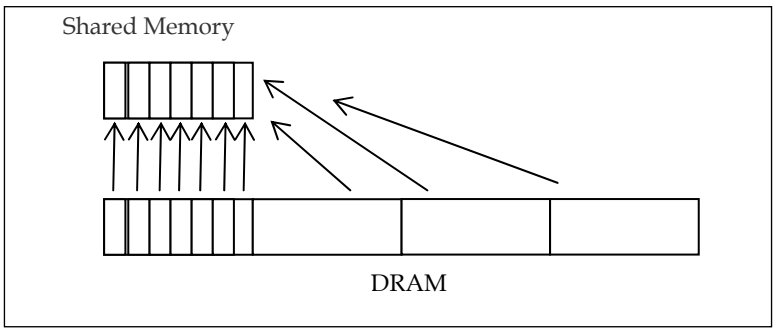

Рис. 2. Замена операций с DRAM на операции с SM

В начале статьи было сделано допущение, что количество частиц меньше или равно количеству нитей. Пусть теперь частиц много больше 10^4. Графическая карта способна поддерживать в рабочем состоянии миллионы нитей, которым можно поставить во взаимно-однозначное соответствие частицы, но такой способ ведет к лишним накладным расходам. В случае, когда количество частиц больше эффективного количества нитей (count), поставим каждой нити в соответствие несколько частиц. Если частица имеет номер i, то ей в соответствие будут поставлены $i - 1$, $\text{count} + i - 1$, $2 \cdot \text{count} + i - 1$ и т. д. В этом случае алгоритм взаимодействия между CPU и GPU не меняется, и работа с shared memory проходит по тем же правилам, за исключением того, что во время расчета нить будет пересчитывать несколько частиц, а не одну.

Несколько графических сопроцессоров на одном узле

При переходе к 3D-моделированию и увеличении количества частиц ресурсов одного графического ускорителя становится недостаточно. Представим алгоритм для расчета на нескольких GPU в рамках одного физического узла.

Создадим дополнительные потоки на CPU для работы с несколькими графическими ускорителями:

— *управляющий поток* отвечает за инициализацию ресурсов, запись данных, синхронизацию и связь с другими узлами через MPI;

— *GPU-поток* — код для решения подзадачи на одной GPU;

— *CPU-поток* — код для решения подзадачи на ядре процессора.

Модифицируем рисунок 1 под текущую ситуацию. Массив input — полная копия data и используется так же, как в случае одной GPU. Массив output — та часть данных, которую пересчитывает выбранная GPU. На рисунке 3 массив output равен половине массива input. Массив data расположен в оперативной памяти CPU, отмечен как невыгружаемый в swap, и его адреса спроецированы в адресное пространство DRAM всех видеокарт (нуль-копируемая память).

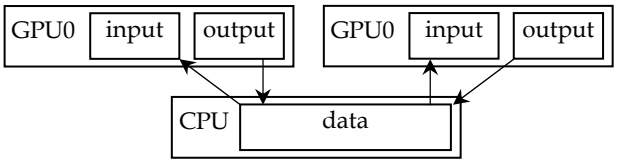

Рис. 3. Модификация данных в памяти

Алгоритм строится по следующей схеме.

1. Опрос устройств узла и выбор типов и количества потоков.
2. Создание потоков.
3. Инициализация графических устройств.
4. Чтение входных данных.
5. Подготовка дополнительной памяти на GPU.
6. Перенесение данных на графические процессоры.
7. Решение задачи на одном шаге.
8. Копирование данных в data.
9. Возврат к пункту 6 или сохранение данных.
10. Уничтожение памяти и потоков.

Неизбежным недостатком алгоритма является необходимость переносить пересчитанные данные в оперативную память для дальнейшего их перераспределения по всем устройствам. В качестве оптимизации применяется нуль-копируемая память, которая ускоряет передачу данных на этапе расчета адресов. Невозможно уйти и от буферного массива input, так как производительность падает в десятки раз. Такой алгоритм показывает ускорение в полтора раза при использовании двух GPU по сравнению с алгоритмом, использующим одну GPU. Алгоритм эффективен, если размер всех данных меньше или равен размеру оперативной памяти GPU, что вполне соответствует реалиям в случае решения задачи взаимодействия спутника и плазмы методом молекулярной динамики.

Заключение

Можно сделать вывод, что задача взаимодействия спутника и плазмы для эффективного расчета требует больших вычислительных мощностей при относительно небольших затратах памяти. Основные проблемы при реализации алгоритма, с которыми приходится сталкиваться, — это постоянное поддержание всего объема данных для каждого вычислителя в актуальном состоянии, а также синхронизация и минимизация обменов. Выбор графических ускорителей для расчета и позволяет сфокусировать как можно большее количество вычислителей на одном узле. Проблемы синхронизации решаются на уровне потоков процессора, а задача минимизации обменов — ограниченной пересылкой актуальных данных и хранением рассчитанных данных в общей для всех невыгружаемой памяти. Также значительное ускорение дает эффективное использование управляемого кэша (shared memory) пото-

ковых мультипроцессоров. Полученный алгоритм сложен в реализации, но наиболее эффективно использует вычислительные ресурсы для решения задачи, легко масштабируется для гибридных вычислений и имеет потенциал роста при использовании MPI+CUDA+CPU threads гибридной модели программирования.

Автор выражает благодарность д-ру физ.-мат. наук Л.В. Зинину за полезное обсуждение результатов работы.

Работа выполнена при финансовой поддержке РФФИ по проекту № 15-01-00369а.

Список литературы

1. *Альперт Я.Л., Гуревич А.В., Питаевский Л.П.* Искусственные спутники в разреженной плазме. М., 1964.

2. *Зинин Л.В., Гальперин Ю.И., Гладышев В.А. и др.* Математическая модель измерений тепловой анизотропной плазмы энерго-масс-угловыми спектрометрами ионов на заряженном спутнике // Космические исследования. 1995. Т. 33, № 6. С. 563—571.

3. *Bouhram M., Dubouloz N., Hamelin M. et al.* Electrostatic interaction between Interball-2 and the ambient plasma. 1. Determination of the spacecraft potential from current calculations // Ann. Geophys. 2002. Vol. 20, N 3. P. 365—376.

4. *Hamelin M., Bouhram M., Dubouloz N. et al.* Electrostatic interaction between Interball-2 and the ambient plasma. 2. Influence on the low energy ion measurements with Hyperboloid // Ann. Geophys. 2002. Vol. 20, N 3. P. 377—390.

5. *Зинин Л.В., Гальперин Ю.И., Григорьев С.А. и др.* Об измерениях эффектов поляризационного джета во внешней плазмосфере // Космические исследования. 1998. Т. 36, № 1. С. 42—52.

6. *Гальперин Ю.И., Гладышев В.А., Козлов А.И. и др.* Электромагнитная совместимость научного космического комплекса АРКАД-3. М., 1984.

7. *Ридлер В., Торкар К., Веселов М. В. и др.* Эксперимент РОН по активному регулированию электростатического потенциала космического аппарата // Космические исследования. 1998. Т. 36, № 1. С. 53—62.

8. *Torkar K., Veselov M.V., Afonin V.V. et al.* An experiment to study and control the Langmuir sheath around INTERBALL-2 // Ann. Geophys. 1998. Vol. 16. P. 1086—1096.

9. *Zinin L., Grigoriev S., Rylina I.* The models of electric field distributions near a satellite // Proceedings of the conference in memory of Yuri Galperin / eds. L.M. Zelenyi, M.A. Geller, J. H. Allen. CAWSES Handbook-001, 2004. P. 76—83.

10. *Котельников В.А., Котельников М.В., Гидаспов В.Ю.* Математическое моделирование обтекания тел потоками столкновительной и бесстолкновительной плазмы. М., 2010.

11. *Рылина И.В., Зинин Л.В., Григорьев С.А. и др.* Гидродинамический подход к моделированию распределения тепловой плазмы вокруг движущегося заряженного спутника // Космические исследования. 2002. Т. 40, № 4. С. 395—405.

12. *Зинин Л.В., Ишанов С.А., Шарамет А.А. и др.* Моделирование распределения ионов вблизи заряженного спутника методом молекулярной динамики. 2-D приближение // Вестник Балтийского федерального университета им. И. Канта. 2012. Вып. 10. С. 53—60.

13. *Шарамет А.А., Зинин Л.В. Ишанов С.А. и др.* 2D моделирование ионной тени за заряженным спутником методом молекулярной динамики // Вестник Балтийского федерального университета им. И. Канта. 2013. Вып. 10. С. 26—30.

14. *Шарамет А. А., Зинин Л. В.* Влияние относительной скорости спутника и плазмы на ионную тень заряженного спутника при 2D моделировании методом молекулярной динамики // Высокопроизводительные вычисления — ма-

тематические модели и алгоритмы : материалы II Международной конференции, посвященной Карлу Якоби. Калининград, 3—5 октября 2013 г. Калининград, 2013. С. 226—227.

15. *Зинин Л.В., Шарамет А.А., Ишанов С.А. и др.* Моделирование траекторий электронов и ионов тепловой плазмы в электрическом поле спутника методом молекулярной динамики // Вестник Балтийского федерального университета им. И. Канта. 2014. № 10. С. 47—52.

Об авторе

Александр Александрович Шарамет — асп., Балтийский федеральный университет им. И. Канта, Калининград.

E-mail: alexsharamet@gmail.com

About the author

Alexandr Sharamet, PhD student, I. Kant Baltic Federal University, Kaliningrad.

E-mail: alexsharamet@gmail.com