РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ЦИКЛИЧЕСКОЙ ПРОГОНКИ НА ГРАФИЧЕСКОМ ВЫЧИСЛИТЕЛЬНОМ УСТРОЙСТВЕ

Л.В.Логанова

Самарский государственный аэрокосмический университет им. академика С.П. Королева

e-mail: tk@smr.ru

We discuss the realization of a cyclic sweep algorithm that enables the execution on a hybrid computing system. Realization variants allowing the algorithm-based computation on several graphics processing units (GPUs) are considered. The use of the GPU has shown a 10-fold computation speedup when compared with the central-processor-based computing.

Введение

Применение параллельных вычислений в математическом моделировании открывает новые возможности, позволяющие исследовать физические процессы на более длительных временных и более обширных пространственных областях. Особой популярностью в нанооптике и нанофотонике пользуется Finite-Difference Time-Domain (FDTD) метод моделирования распространения электромагнитного излучения в рамках строгой теории дифракции. Относящиеся к этому методу разностные схемы Zheng, Chen, Zhang [1] и известные параллельные варианты FDTD [2] пробудили интерес автора к синтезу эффективных алгоритмов, ориентированных на графические вычислительные устройства. Действительно, современное графическое вычислительное устройство (GPU) способно выполнять большое количество арифметических операций параллельно. К тому же программно-аппаратная архитектура NVIDIA CUDA позволяет использовать для программирования приложений на GPU стандартный язык программирования C с расширениями[3].

Настоящая работа посвящена исследованию алгоритма решения сеточных уравнений схем Zheng, Chen, Zhang [1], основанному на методе циклической прогонки [4]. Известно, что алгоритмы, основанные на методе прогонки, по сравнению с методами циклической редукции и декомпозиции области характеризуется меньшим объемом коммуникаций и арифметических операций[5]. При этом ограничение на масштабируемость алгоритма может быть снято в случае многомерной области. Известны эффек-

тивные параллельные реализации алгоритма циклической прогонки на кластерных вычислительных системах [6]. В настоящей работе представлена реализация алгоритма с применением технологии CUDA. Вычисления по алгоритму при этом выполняются на центральных процессоре (ЦПУ) и на GPU, т.е. на так называемой гибридной вычислительной системе. Интересной, по мнению автора, является возможность реализации алгоритма с применением нескольких графических вычислительных устройств, установленных на одном вычислительном узле.

Алгоритм метода циклической прогонки

Предположим, что для решения некоторой задачи математической физики используется двумерная сеточная область. Воспользовавшись для ее решения методом покоординатного расщепления, приходим к необходимости решения совокупности СЛАУ следующего вида: $A_k y_k = d_k$ (1), где $y_k, d_k \in R^M$, $k = \overline{1, N}$,

$$A_k = \begin{pmatrix} c_0 & b_0 & ... & ... & ... & 0 & a_0 \\ a_1 & c_1 & b_1 & ... & ... & 0 & 0 \\ 0 & ... & ... & ... & ... & ... & 0 \\ 0 & ... & ... & ... & ... & ... & 0 \\ b_{M-1} & 0 & ... & ... & ... & a_{M-1} & c_{M-1} \end{pmatrix}$$

При этом каждой строке и столбцу сеточной области соответствует одна система (1). Следовательно, решение всех разностных уравнений схемы (на одном временном слое, если схема нестационарная) связано с решением M уравнений по N строкам сеточной области и N уравнений по M столбцам. Для решения СЛАУ (1) воспользуемся методом циклической прогонки [4]. Представим решение в виде линейной комбинации сеточных функций: $y_i = u_i + y_0 v_i, 0 \le i \le M-1$ (2).

Для отыскания значений промежуточных сеточных функций (u_i, v_i), прогоночных коэффициентов $(\alpha, \beta, \gamma$)воспользуемся соответствующими формулами [4],

$\alpha_2 = \frac{b_1}{c_1}, \beta_2 = \frac{d_1}{c_1}, \gamma_2 = \frac{a_1}{c_1}$ (3)

$\alpha_{i+1} = \frac{b_i}{c_i - a_i\alpha_i}, \beta_{i+1} = \frac{d_i + a_i\beta_i}{c_i - a_i\alpha_i}, \gamma_{i+1} = \frac{a_i\beta_i}{c_i - a_i\alpha_i}, 2 \le i \le M-1$ (4)

$u_{M-1} = \beta_M, v_{M-1} = \alpha_M + \gamma_M$ (5)

$u_i = \alpha_{i+1}u_{i+1} + \beta_{i+1}, v_i = \alpha_{i+1}v_{i+1} + \gamma_{i+1}, 1 \le i \le M-2$ (6)

$y_0 = \frac{d_0 + a_0 u_{M-1} + b_0 u_1}{c_0 - a_0 v_{M-1} - b_0 v_1}$ (7)

Аналогично производятся циклические прогонки по столбцам. Отметим, что при реализации данного алгоритма оптимальной схемой хранения матрицы является представление ее в виде векторов (a, b, c), каждый из которых соответствует одной из диагоналей исходной матрицы: $a = (a_0, a_1, ..., a_{M-1}), b = (b_0, b_1, ..., b_{M-1}), c = (c_0, c_1, ..., c_{M-1})$.

При этом формулы (2–7) не изменяются.

Реализация алгоритма на системе с одним графическим вычислительным устройством

Первоначальный вариант реализации алгоритма циклической прогонки, в которой вызывалось одно ядро, и каждая нить выполняла вычисления по алгоритму (формулы (2-7)), оказалась не эффективной даже при больших размерах сеточной области. Это было связано с тем, что размер векторов, соответствующих диагоналям был таким большим (для совокупности систем размер векторов увеличивался в N раз для строк и M - для столбцов), что при передаче их с ЦПУ на GPU время пересылки оказывалось незначительно меньше, чем время вычислений, производимых на GPU. Высокими оказались требования к памяти GPU. В связи с чем была предпринята попытка декомпозиции вычислений. В этом случае алгоритм может быть следующим:
1.Задание характеристик ядра. Передача данных на GPU($a, b, c, d \in R^{M \cdot N}$);
2.Вызов ядра. Вычисление прогоночных коэффициентов $\alpha_i, \beta_i, \gamma_i$ (3,4);
3.Удаление a, b, c, d из памяти GPU;
4.Вызов ядра. Нахождение значений промежуточных сеточных функций u_i, v_i (5,6);
5.Удаление векторов α, β, γ из памяти GPU. Передача значений промежуточных сеточных функций u,v на ЦПУ;
6.Вычисление y_0 на ЦПУ (7). Передача y_0 на GPU;
7.Вызов ядра. Определение искомых значений сеточных функций y_i (2) и пересылка на ЦПУ.
В данной реализации применялась глобальная память, время вычислений по алгоритму уменьшилось в 3 раза.

Известно, что правильное использование разделяемой памяти играет огромную роль при написании эффективных программ для GPU[7]. Она является одним из самых быстрых видов памяти, размещена в самом мультипроцессоре, доступна для чтения и записи всем нитям блока.

Использование разделяемой памяти при проведении вычислений на GPU позволило значительно увеличить скорость вычислений. При этом структура программы изменениям не подверглась.

Исследования алгоритмов (с применением глобальной и разделяемой памяти) проводились на гибридной вычислительной системе с видеокартой GeForce GTX 470 , CPU Intel Core 2 Duo E8500 3.16 ГГц, операционной системой Microsoft Windows XP с установленным драйвером NVIDIA CUDA Version 2.3.

На рисунке 1 приведены графики зависимости отношения времени выполнения вычислений по алгоритму на ЦПУ(T_{CPU}) ко времени выполнения вычислений на GPU(T_{GPU}) от размеров сеточной области (для M=500) и применения различных видов памяти GPU. Естественно, реализация алгоритма с использованием разделяемой памяти поз-

воляет почти в два раза уменьшить время выполнения вычислений по алгоритму.

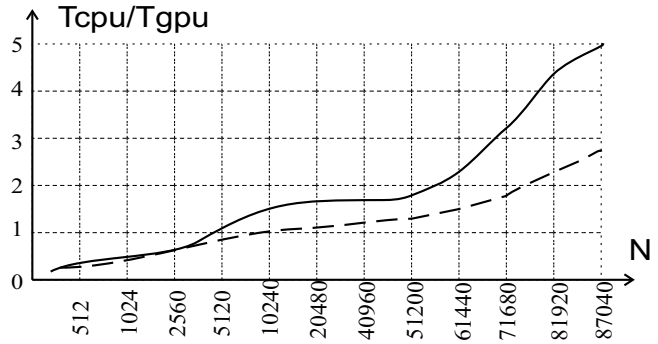

Рис. 1. Зависимость T_{CPU}/T_{GPU} от размеров сеточной области M=500 (непрерывная линия – с применением разделяемой памяти, пунктирная – глобальной памяти)

На рисунке 2 показана зависимость отношения T_{CPU}/T_{GPU} от размеров сеточной области. Рассматривались случаи, когда число уравнений равнялось 300, 500, 700, 1000 для одних и тех же N(число строк). В данном случае использовалась разделяемая память. Из графиков видно, что чем больше объем производимых вычислений, тем быстрее растет отношение T_{CPU}/T_{GPU}. Ускорение вычислений больше 1 достигается при M=1000 уже для N=2560, а при M=500,700 только при N=5120. Несмотря на то, что

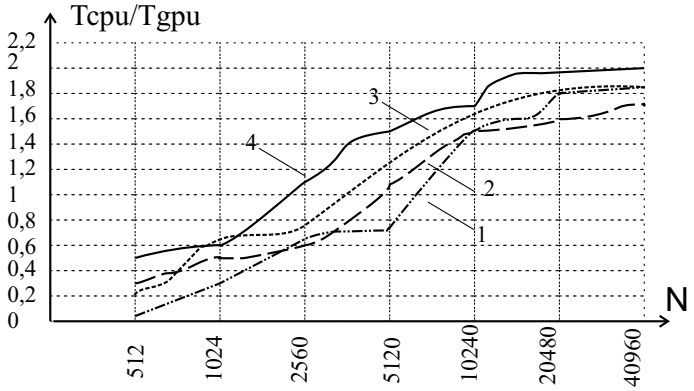

Рис. 2. Зависимость ускорения вычислений от размерности сеточной области (1 – M=300, 2 – M=500, 3 – M=700, 4 – M=1000)

прямые методы с узкой лентой эффективно не векторизуются [8], предложенная реализация алгоритма циклической прогонки на графическом вычислительном устройстве позволила уменьшить время выполнения вычислений по алгоритму в 5 раз (рис. 1).

Реализация алгоритма на системе с двумя графическими вычислительными устройствами

Для использования нескольких GPU на одном вычислительных узле при выполнении вычислений необходимо одновременно запустить несколько потоков ЦПУ. Каждый из них выделит данные, необходимые ему для обработки, и выберет GPU для вычислений. Передача данных потокам ЦПУ реализуется средствами OpenMP [9]. Алгоритм при этом не требует изменений. Каждый GPU выполняет вычисления по алгоритму (формулы (2-7)) для половины строк, т.е. первый обрабатывает строки с 1 по N/2, второй - с N/2+1 по N (в отличие от N циклических прогонок в случае использования одного GPU). Результаты проведенных вычислительных экспериментов представлены на рисунках 3,4 . Представленные графики (рис.3)позволяют сделать вывод, как и в

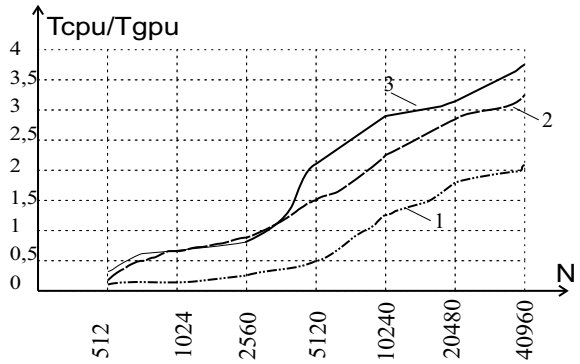

Рис. 3. Зависимость ускорения вычислений от размерности сеточной области (1 – M=300, 2 – M=500, 3 – M=1000)

случае применения 1 GPU, что отношение T_{CPU}/T_{GPU} растет быстрее с ростом M. Графики, приведенные на рисунке 4, позволяют сравнить ускорение вычислений в случае применения одного или двух GPU. На графиках видно, что разница их значений

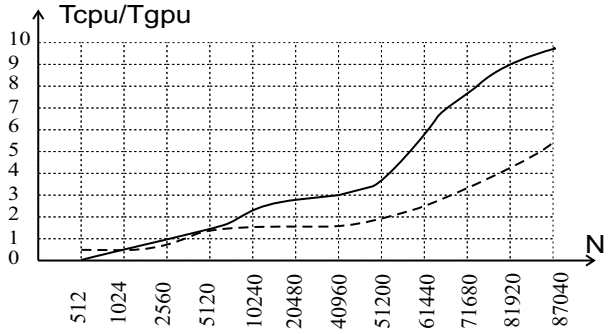

Рис. 4. Зависимость T_{CPU}/T_{GPU} от размеров сеточной области M=500 (непрерывная линия – 2 GPU, пунктирная – 1 GPU)

возрастает с ростом размера сеточной области. Начиная с N=71680, значение ускоре-

ния вычислительного процесса, выполняемого на 2 GPU, оказывается в 2 раза больше. Таким образом, максимальное ускорение вычислительного процесса при применении 2 GPU для вычислений составило 9,95 раз по сравнению с вычислениями проводимыми только на ЦПУ.

Заключение

В настоящей работе предложены реализации алгоритма циклической прогонки для выполнения порождаемых ими вычислительных процессов на гетерогенной вычислительной системе, состоящей из одного ЦПУ и нескольких графических вычислительных устройств. Полученные в результате проведенных вычислительных экспериментов ускорения (для случаев с 1 и 2 GPU) позволяют говорить о их эффективности. Представляется целесообразным исследование возможности применения декомпозиции, которая предполагает проведение зависимых вычислений по алгоритму на нескольких графических вычислительных устройствах.

Список литературы

[1] Taflove A., Hagness S. Computational Electrodynamics: The Finite-Difference Time-Domain Method: 3 nd.ed. – Boston:Arthech House Publishers, 2005.- 852 p.

[2] Wenhua Yu, Raj Mittra, Tao Su, Yongjun Liu, Xiaoling Yang Parallel finite-difference time-domain method.—(Artech House electromagnetic analysis series), 2006. 274 с.

[3] NVIDIA CUDA. Reference Manual. February, 2010. Version 3.0.

[4] Самарский А. А., Николаев Е.С. Методы решения сеточных уравнений. М.: Наука, 1978, 561 с.

[5] Головашкин Д. Л. Параллельные алгоритмы решения сеточных уравнений трехдиагонального вида, основанного на методе встречных прогонок. // Математическое моделирование. 2005. Т. 17, № 11. С. 118–128.

[6] Головашкин Д. Л., Филатов М.В. Параллельные алгоритмы метода циклической прогонки.// Компьютерная оптика. 2005. №. 27, С. 123–130.

[7] Боресков А. В., Харламов А. А. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. - 232 с.: ил.

[8] Дж. Ортега Введение в параллельные и векторные методы решения линейных систем: Пер. с англ. -М: Мир, 1991. - 367 с.,ил.

[9] Антонов А. С.Параллельное программирование с использованием технологии OpenMP: Учебное пособие. – М.: Изд-во МГУ, 2009. – 77 с.