Распараллеливание решения сеточных уравнений на квазиструктурированных сетках с использованием графических ускорителей[1]

Медведев А.В.[*], Свешников В.М.[**], Турчановский И.Ю.[*]

[*]Институт вычислительных технологий СО РАН, Томский филиал, tur@hcei.tsc.ru

[**]Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, victor@lapasrv.sscc.ru

Аннотация

Проведено распараллеливания метода последовательной верхней релаксации для решения сеточных уравнений, возникающих при аппроксимации краевых задач на квазиструктурированных сетках, с использованием графических ускорителей GPU. Показано, что быстродействие программы для GPU в рамках текущей реализации по отношению к аналогичной программе на CPU (центральный процессор) увеличивается в 17.6 раз для типа данных $float$ и в 9.5 раза для типа данных $double$.

Введение

При решении краевых задач на квазиструктурированных сетках [1] значительный объем вычислений составляет решение задач в подобластях, являющихся сеточными макроэлементами. Цель настоящей работы состоит в распараллеливании решения сеточных уравнений в подобластях с использованием графических ускорителей GPU, поддерживающих архитектуру CUDA. В качестве метода решения применяется метод последовательной верхней релаксации [2]. Такой выбор не случаен, так как в процессе выполнения работы был обнаружен внутренний параллелизм данного метода, который позволил эффективно применять ускорители GPU. Были проведены численные эксперименты, которые показали, что быстродействие расчетов на графическом ускорителе по сравнению с аналогичными расчетами на центральном процессоре, увеличивается в 17.6 раз для типа данных $float$ и в 9.5 раз для типа $double$.

Постановка краевой задачи

В замкнутой двумерной области $\overline{G} = G \cup \Gamma$ с границей Γ требуется решить краевую задачу:

$$\Delta u = g_1,\ lu|_{\Gamma} = g_2,$$

где $u = u(T)$ – искомая функция; $g_1 = g_1(T)$, $g_2 = g_2(T)$ – заданные функции ($T = (x, y)$ – текущая точка; x, y – декартовы или цилиндрические координаты); Δ – оператор Лапласа; l – оператор граничных условий. Решение данной задачи ищется на квазиструктурированной сетке.

Построение квазиструктурированных сеток

Вокруг расчетной области $\overline{G}$ описывается прямоугольник $R = \{0 \le x \le D_x, 0 \le y \le D_y\}$, где D_x, D_y – заданы. В R строится равномерная *макросетка*: $\Omega_H = \{X_I = IH_x; Y_J =$

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 12-01-00076) и СО РАН (интеграционные проекты №№ 104, 126, 130)

$JH_y; I = \overline{0, N_x};\ J = \overline{0, N_y}; H_x = \frac{D_x}{N_x}; H_y = \frac{D_y}{N_y}\Big\}$, где N_x, N_y – заданные числа, H_x, H_y – шаги, которые намного превосходят шаги сетки, на которой аппроксимируется исходная задача.

Таким образом, исходная область фактически подвергается декомпозиции на непересекающиеся подобласти.

Все подобласти делятся на три типа: внутренние, содержащие только точки G, граничные, содержащие точки G и Г, а также внешние, не содержащие точек G. В дальнейшем внешние подобласти исключаются из расчетов.

Точки пересечения координатных линий образуют *макроузлы*. Координатные линии макросетки являются границами сопряжения подобластей или *интерфейсом*.

После того как построена макросетка, внутри каждой подобласти строится своя сетка (*микросетка*) $\overline{\Omega}_{h,k} = \{x_{i_k} = X_I + i_k h_{x,k},\ y_{j_k} = Y_J + j_k h_{y,k},\ i_k = \overline{0, n_{x,k}},\ j_k = \overline{0, n_{y,k}}\}$, где $n_{x,k}, n_{y,k}$ – заданные целые числа, с шагами $h_{x,k} = \frac{X_{I+1} - X_I}{n_{x,k}};\ h_{y,k} = \frac{Y_{J+1} - Y_J}{n_{y,k}}$. Подчеркнем, что величины $n_{x,k}, n_{y,k}$, которые определяют плотность узлов в подсетках, задаются в зависимости от особенностей решаемой задачи, то есть подсетки могут быть несогласованными.

На интерфейсе строится своя сетка ω_h, которая состоит из узлов подсеток $\overline{\Omega}_{h,k}$.

На рис.1 изображен фрагмент квазиструктурированной сетки, состоящий из четырех подобластей (черные кружки обозначают макроузлы).

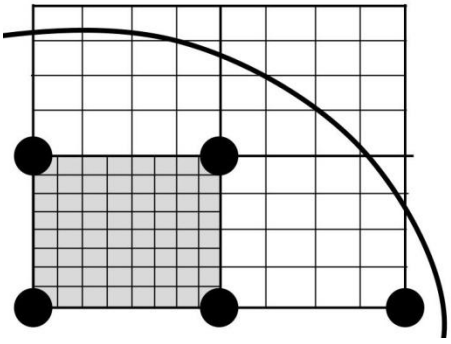

Рис. 1 Фрагмент квазиструктурированной сетки

Решение сеточных уравнений в подобластях

Решение сеточных уравнений в подобластях, аппроксимирующих исходное уравнение, будем проводить методом последовательной верхней релаксации (ПВР). Чтобы пояснить внутренний параллелизм данного метода рассмотрим простейший случай: решение 5-ти точечных разностных уравнений, аппроксимирующих уравнение Лапласа на квадратной сетке. Тогда метод последовательной верхней релаксации дается следующей формулой:

$$u_{ij}^n = w * \frac{\left(u_{i-1,j}^n + u_{i,j-1}^n + u_{i+1,j}^{n-1} + u_{i,j+1}^{n-1}\right)}{4} + (1 - w) * u_{ij}^n$$

где n – номер итерации; w – релаксационный параметр; u_{ij} – значение искомой функции в узле с сеточными координатами (i, j).

Последовательный ход вычислений этого метода, который характерен для расчетов на CPU (Central Processing Unit – центральный процессор), можно наглядно изобразить следующим образом (см. рис.2):

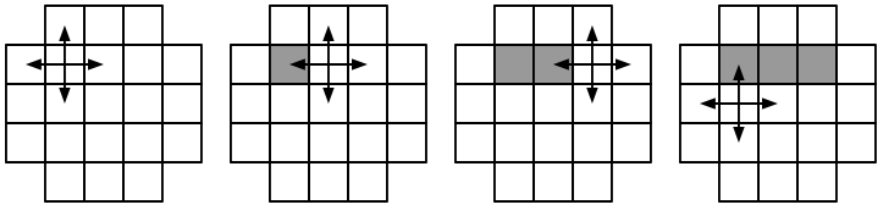

Рис.2.Последовательные вычисления в методе ПВР

Если изменить порядок вычислений и проводить их по диагонали сеточной подобласти (см. рис. 3), то становится очевидным, что значения на одной диагонали можно вычислять параллельно

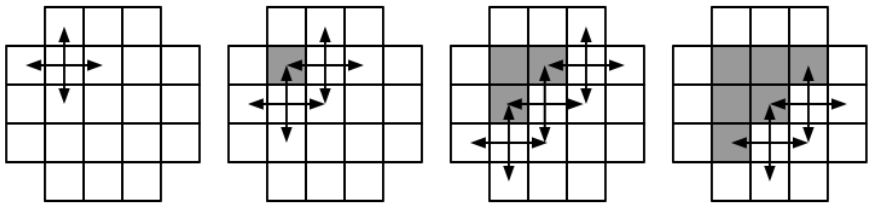

Рис.3.Параллельные вычисления в методе ПВР

Этот факт был положен в основу распараллеливания с привлечением GPU.

Технология проведения расчетов

Для нахождения решения исходной краевой задачи на квазиструктурированной сетке применяется метод декомпозиции расчетной области на непересекающиеся подобласти. Сам метод декомпозиции и технология его применения к квазиструктурированным сеткам изложены в работах [3, 4]. Суть рассматриваемого подхода заключается в проведении итераций по подобластям или макроэлементам квазиструктурированной сетки. На каждой такой итерации необходимо решать краевые задачи Дирихле во внутренних и граничных подобластях. Распараллеливание самого итерационного процесса по подобластям проводится на CPU в рамках системы MPI [5]. Распараллеливание решения краевых задач осуществляется на GPU в рамках системы CUDA[6]. Именно этот вопрос находится в центре внимания данной статьи. Согласно диаграмме проведения расчетов, изображенной на рис. 3 каждый узел, лежащий на диагонали, может быть рассчитан параллельно. Поэтому было принято следующее отображение: один узел – одно ядро GPU. Диагонали перебираются последовательно. В целом технология проведения расчетов выглядит следующим образом:

1. На CPU на каждом шаге итерационного процесса по подобластям вычисляются значения искомой функции в узлах сетки ω_h на интерфейсе.
2. Эти значения копируются в память GPU.
3. На GPU методом ПВР согласно установленному распараллеливанию и принятому отображению решаются краевые задачи в подобластях.
4. Полученные значения копируются в память CPU.
5. Пересчитываются значения на интерфейсе и описанный процесс повторяется, начиная с шага 2, пока он не сойдется с заданной точностью.

Численные эксперименты

Рассматривалась модельная краевая задача Дирихле для уравнения Лапласа с известным аналитическим решением. Целью проводимых экспериментов было установить ускорение вычислений распараллеленного метода ПВР на GPU по отношению к последовательным расчетам на CPU тем же методом. Для этого была написана программа на языке программирования CUDA C. Расчетные эксперименты проводились на оборудовании Intel Core i5, NVIDIA GPU Tesla C2070 в среде linux CentOS 6.2.

Квазиструктурированная сетка состояла из подсеток, каждая из которых содержат $32 * 32$ узла, что обусловлено тем, что на каждом мультипроцессоре именно 32 дискретных вычислителя, а потоки (нити) исполняются группами по 32. Макросетка имела параметры $N_x = N_y = N$, причем N изменялось в пределах $1,..,100$.

Ниже приведены графики зависимости ускорения от N (на рис. 4 – для типа данных $float$, на рис.5 – для $double$). В силу аппаратных особенностей GPU, для типа данных $float$ максимально допустимой точностью является $\varepsilon = 1.0e-5$, а для $double$ – $\varepsilon = 1.0e-14$. Эти значения, в дальнейшем, и были взяты в качестве условия остановки итерационного процесса верхней релаксации.

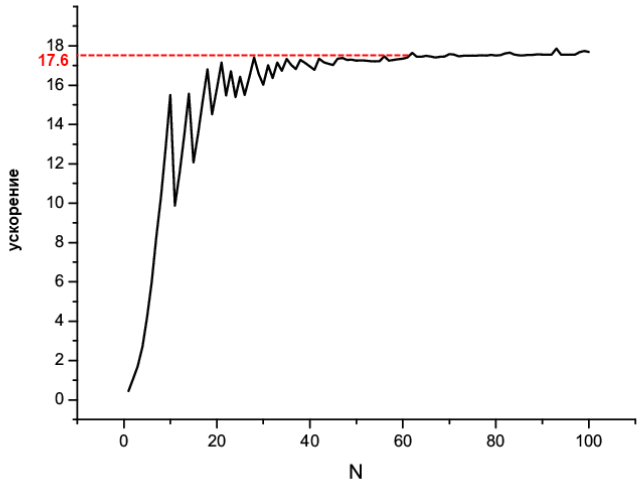

Рис. 4. График зависимости ускорения от количества подобластей для типа данных $float$.

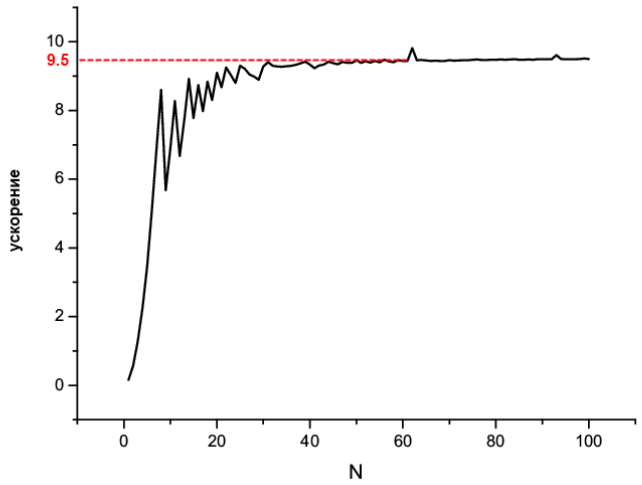

Рис. 5. График зависимости ускорения от количества подобластей для типа данных $double$.

Поведение графиков обусловлено тем, что, начиная с некоторого N, на GPU становятся задействованными практически все процессорные ресурсы, т.е. каждый мультипроцессор загружен на ≈100%, после чего значение ускорения стабилизируется. Если сравнивать графики между собой, то видно, что значения ускорения разнятся примерно в 2 раза. Это объясняется тем, что операций над типами $double$ работают с уменьшенной в 2 раза тактовой частотой.

Ниже в таблице 1 приводятся времена расчетов с использованием GPU для $N = 256$ и $n = 32$, то есть количество узлов, соответствующих внутренним подобластям, равно $256 * 256 * 32 * 32 = 4 * 256^3 \approx \approx 67$млн.

Тип данных	float	double	
ε	1.0e-5	1.0e-5	1.0e-14
Время (сек.)	9.69	19.37	54.44

Таблица 1 Время счета на GPU (в секундах) для $N = 256$.

Заключение

В настоящей работе получены следующие основные результаты

1. Проведена адаптация метода последовательной верхней релаксации для распараллеливания на GPU.

2. Разработана программа на языке CUDA C, решающая краевую задачу Дирихле для уравнения Лапласа на квадрате размерами $N * N$ подобластей, каждая из которых содержит $n * n$ узлов при условии, что значения на интерфейсе уже заданы.

3. Получено, что быстродействие программы для GPU в рамках текущей реализации по отношению к аналогичной программе на CPU, увеличивается в 17.6 раз для типа данных $float$ и в 9.5 раза для типа данных $double$.

Литература

[1]. Беляев Д.О., Свешников В.М. Построение квазиструктурированных локально-модифицированных сеток для решения задач сильноточной электроники
Вестник ЮУрГУ. 2012. № 40(299). С. 118 – 128.

[2]. Ильин В. П. Методы конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск: Издательство Института математики, 2000. 345 с.
[3]. Свешников В.М. Построение прямых и итерационных методов декомпозиции. Сиб. Журн. Идустр. Матем.. 2009. Т.12, № 3(39). С. 99 – 109.
[4]. Свешников В.М. Параллельные технологии решения краевых задач на квазиструктурированных сетках: 2010. Труды Международной суперкомпьютерной конференции Научный сервис в сети Интернет. М.: МГУ. С. 238 – 241.
[5]. Корнеев В.Д. Параллельное программирование в MPI. 2002. Новосибирск. ИВМиМГ СО РАН.
[6]. NVIDIA. CUDA C Best Practices Guide. 2012.