УДК 519.63 doi 10.26089/NumMet.v17r107

ТЕХНОЛОГИИ РАСПАРАЛЛЕЛИВАНИЯ РЕШЕНИЯ ТРЕХМЕРНЫХ КРАЕВЫХ ЗАДАЧ НА КВАЗИСТРУКТУРИРОВАННЫХ СЕТКАХ В ГИБРИДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЕ CPU+GPU

И. А. Климонов[1], В. Д. Корнеев[2], В. М. Свешников[3]

При распараллеливании решения трехмерных краевых задач на квазиструктурированных сетках методом декомпозиции расчетной области на подобласти, сопрягаемые без наложения, наиболее трудоемкой вычислительной процедурой является решение краевых подзадач в подобластях. Использование параллелепипедальных квазиструктурированных сеток дает возможность применить для этих целей быстросходящиеся методы переменных направлений. Распараллеливание итерационного процесса по подобластям проводится на CPU в системе MPI, а для решения подзадач в настоящей статье предлагается использовать графические ускорители GPU. Приводятся экспериментальные исследования применения графических ускорителей при решении подзадач методом Писмана–Речфорда. Даются экспериментальные оценки ускорения распараллеливания в гибридной вычислительной среде CPU+GPU по сравнению с расчетами только на CPU.

Ключевые слова: краевые задачи, методы декомпозиции области, уравнение Пуанкаре–Стеклова, квазиструктурированные сетки, метод Писмана–Речфорда, графические ускорители.

1. Введение. Основой распараллеливания решения трехмерных краевых задач, рассматриваемого в настоящей работе, является метод декомпозиции расчетной области [1, 2] на параллелепипедальные подобласти, сопрягаемые без наложения. В каждой такой подобласти строится своя равномерная параллелепипедальная подсетка. Совокупность подсеток образует квазиструктурированную сетку. Ее достоинством является, с одной стороны, простота использования и, с другой стороны, адаптивность к решению за счет регулировки плотности узлов подсеток. Сшивка решений в подобластях осуществляется путем прямой аппроксимации и решения уравнения Пуанкаре–Стеклова на границе сопряжения подобластей (интерфейсе), что приводит к итерационному процессу, на каждом шаге которого необходимо решать краевые подзадачи. Отметим, что для двумерной постановки данный подход рассматривался в работах [3–5]. Подсетки группируются в объединения, содержащие приблизительно одинаковое число узлов, для балансировки загрузки процессоров многопроцессорной суперЭВМ. Строится отображение "одно объединение–один процессор", согласно которому вычислительный процесс распараллеливается на CPU в системе MPI [6]. Наибольшая вычислительная нагрузка при этом приходится на решение краевых подзадач.

В настоящей статье предлагается и экспериментально исследуется способ ускорения таких вычислений, а именно применение для этих целей графических ускорителей GPU, т.е. решение всей задачи по сути дела проводится в гибридной вычислительной среде CPU+GPU. В качестве метода решения подзадач выбран трехмерный аналог метода Писмана–Речфорда [7], который легко распараллеливается, так как состоит из независимых прогонок по различным направлениям. Ниже рассматриваются технологии его реализации в системе CUDA [8] и даются результаты численных экспериментов, показывающие значительное (более 60 раз) ускорение вычислений по сравнению с расчетами только на CPU.

2. Постановка задачи и основы алгоритма ее решения. Пусть в замкнутой трехмерной области $\overline{G} = G \cup \Gamma$ с границей Γ требуется решить краевую задачу

$$\Delta u = g_1, \quad l\,u\big|_\Gamma = g_2. \tag{1}$$

Здесь $u = u(T)$ — искомая функция; $g_1 = g_1(T)$, $g_2 = g_2(T)$ — заданные функции ($T = (x, y, z)$ — текущая точка, где x, y, z — декартовы координаты); Δ — оператор Лапласа, l — оператор граничных

[1] Новосибирский государственный университет, механико-математический факультет, ул. Пирогова, д. 2, 630090, Новосибирск; студент, e-mail: ilya.klimonov@gmail.com

[2] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; ст. науч. сотрудник, e-mail: korneev@ssd.sscc.ru

[3] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; зав. лабораторией, e-mail: victor@lapasrv.sscc.ru

условий. Рассматриваются граничные условия Дирихле, Неймана, а также смешанные краевые условия. Предполагается, что граница Г и функции g_1, g_2 таковы, что существует единственное решение задачи (1), обеспечивающее гладкость, достаточную для проведения дальнейших рассуждений.

Построим в расчетной области $\overline{G}$ структурированную равномерную макросетку $\overline{\Omega}_H$ с шагами, намного превышающими максимальный шаг результирующей сетки, на которой ищется решение исходной задачи. Тем самым мы проведем декомпозицию области $\overline{G}$ на непересекающиеся подобласти $\overline{G}_m$, $m = \overline{1, M}$, где M — известное целое число. Граница сопряжения подобластей (интерфейс) γ, в свою очередь, разбивается на грани γ_f, ребра γ_e и макроузлы γ_m, являющиеся узлами макросетки $\overline{\Omega}_H$, так, что $\gamma = \gamma_f \cup \gamma_e \cup \gamma_m$. Для дальнейшего удобно ввести объединение $\gamma_{e,m} = \gamma_e \cup \gamma_m$ и область $G_0 = G \backslash \gamma$.

Построим в подобластях $\overline{G}_m$ структурированные равномерные подсетки $\overline{\Omega}_{h,m}$. Объединение этих подсеток составляет квазиструктурированную сетку $\overline{\Omega}_h = \cup_{m=1}^{M} \overline{\Omega}_{h,m}$.

Исходную краевую задачу (1) переформулируем следующим образом: в замкнутой области $\overline{G}$ требуется найти решение уравнения Пуанкаре–Стеклова

$$Fv(T) = 0, \quad T \in \gamma_f, \tag{2}$$

совместно с решением краевых задач

$$\Delta u(T) = g_1(T), \quad T \in \gamma_{e,m}, \tag{3}$$

$$\Delta\, u(T) = g_1(T)\,, \quad l\, u\big|_{\Gamma} = g_2, \quad u\big|_{\gamma} = v, \quad T \in G_0, \tag{4}$$

относительно функций u и v. Здесь оператор F определяется как

$$Fv \equiv \left(\frac{\partial u(v)}{\partial \boldsymbol{n}}\right)^{(+)}_{\gamma_f} - \left(\frac{\partial u(v)}{\partial \boldsymbol{n}}\right)^{(-)}_{\gamma_f}, \tag{5}$$

где v — след функции u на γ (в том числе на γ_f). Решение задачи (2)–(4) будем проводить методом итераций по подобластям, состоящим из следующих этапов.

1. Задается начальное приближение $v_f^{(0)}$ на гранях γ_f.

2. Из уравнения (3) находятся значения функции $u_{e,m}^{(n)} = v_{e,m}^{(n)}$ ($n = 0, 1, \ldots$ – номер итерации) на ребрах и в макроузлах $\gamma_{e,m}$.

3. Из решения краевой задачи (4) с граничными условиями Дирихле $v = v^{(n)} = v_f^{(n)} \cup v_{e,m}^{(n)}$ находятся значения искомой функции на n-м приближении в подобластях.

4. Рассчитываются производные, входящие в выражение (5).

5. Делается очередной $(n+1)$-й шаг итераций для решения уравнения Пуанкаре–Стеклова (2) и находятся значения функции $v_f^{(n+1)}$ на гранях.

6. Если сходимость итерационного процесса достигнута, то находится окончательное решение на ребрах, в макроузлах и в подобластях; если же это не так, то повторяются пункты 2–5.

На квазиструктурированной сетке $\overline{\Omega}_h$ краевая задача (4) методом конечных разностей, конечных элементов или конечных объемов заменяется приближенной задачей

$$\Delta_h\, u_h = g_1, \quad l_h\, u_h\big|_{\Gamma} = g_2, \quad u_h\big|_{\gamma} = v_h, \tag{6}$$

где u_h и v_h — приближенные значения функций u и v, а Δ_h и l_h — аппроксимации оператора Лапласа и оператора граничных условий. Ее решение сводится к решению дискретных подзадач на подсетках $\overline{\Omega}_{h,m}$, что может быть выполнено параллельно. Тем самым мы реализуем на каждой итерации по подобластям этап 3 описанного выше алгоритма.

Для решения подзадач на интерфейсе введем на гранях γ_f сетку ω_f, на ребрах γ_e — сетку ω_e. Для единообразия сетку из макроузлов обозначим через $\omega_m = \gamma_m$. В объединении $\gamma_{e,m}$ будем рассматривать сетку $\omega_{e,m} = \omega_e \cup \omega_m$.

Для нахождения функции v_f заменим в (5) производные приближенными разностными соотношениями и потребуем, чтобы разность приближенных производных в узлах сетки ω_f обращалась в нуль, что приводит к системе линейных алгебраических уравнений [3]

$$Av_f + b = 0, \tag{7}$$

где A — квадратная матрица, а b и v_f — векторы.

Элементы матрицы A и вектора b не известны. Известно лишь действие оператора F_h, аппроксимирующего оператор F, на какую-либо функцию $\widetilde{v}_f$, заданную в узлах ω_f. Это действие определяется формулой

$$F_h\widetilde{v}_f = A\widetilde{v}_f + b. \tag{8}$$

Для вычисления компонент вектора b дадим $\widetilde{v}_f$ пробное значение $\widetilde{v}_f^{(0)} = 0$, решим соответствующую краевую задачу, а затем, вычислив $F_h\widetilde{v}_f^{(0)}$, получим искомый вектор.

Решение системы линейных алгебраических уравнений (7) будем проводить каким-либо итерационным методом в подпространствах Крылова [7]. Замечательным свойством этих методов, которое мы используем, является то, что они не требуют знания элементов матрицы A, а требуют лишь действия A на некий вектор p, что согласно (8) может быть вычислено по формуле $Ap = F_hp - b$.

На каждом шаге крыловского итерационного процесса необходимо решать краевые подзадачи в подобластях, поэтому его называют итерационным процессом по подобластям. Положительным свойством трудоемкой процедуры итераций по подобластям является то, что она допускает параллельную реализацию.

Решение подзадач на ребрах и в макроузлах составляет этап 2 описанного выше алгоритма и проводится на каждой n-й итерации по подобластям. К моменту проведения указанных вычислений значения функции $v_f^{(n)}$ на гранях считаются известными.

На сетке $\omega_{e,m}$ с привлечением узлов сетки ω_f аппроксимируем исходное уравнение (3). Получим систему сеточных уравнений

$$B\widetilde{v} = f \tag{9}$$

с матрицей B относительно функции $\widetilde{v}$, определенной на ребрах и в макроузлах. Матрицу B можно представить в блочном виде $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$.

Соответственно, векторы, входящие в (9), представим в форме $\widetilde{v} = \begin{bmatrix} \widetilde{v}_e \\ \widetilde{v}_m \end{bmatrix}$, $f = \begin{bmatrix} f_e \\ f_m \end{bmatrix}$.

Здесь блоки имеют следующие размерности: $B_{11} : N_e \times N_e$, $B_{12} : N_e \times N_m$, $B_{21} : N_m \times N_e$, $B_{22} : N_m \times N_m$, где N_e — число реберных узлов ω_e, а N_m — число макроузлов $\omega_{m,}$.

Решение системы (9) будем проводить при помощи итерационного процесса, каждый шаг которого состоит из двух полушагов

$$B_{11}\widetilde{v}_e^{\nu+1/2} + B_{12}\widetilde{v}_m^{\nu} = f_e, \quad B_{21}\widetilde{v}_e^{\nu+1/2} + B_{22}\widetilde{v}_m^{\nu+1} = f_m, \quad \widetilde{v}_e^{\nu+1} = \widetilde{v}_e^{\nu+1/2}, \tag{10}$$

где $\nu = 0, 1, \ldots$ — номер итерации.

На первом из них вычисляются значения функции на ребрах при фиксированных значениях в макроузлах, а на втором — корректируются значения в макроузлах при фиксированных значениях на ребрах. На ребрах мы имеем “одномерные” сеточные уравнения, так как известные значения функции $v_f^{(n)}$ на гранях, входящих в аппроксимацию уравнения (3), исключаются и переносятся в правую часть f_e.

Второй полушаг по сути дела состоит в пересчете значений в отдельных макроузлах, которые тоже не связаны друг с другом, а связаны с полученными на первом полушаге значениями на ребрах.

Рассмотрим решение подзадач в подобластях. Запишем подзадачу вида (6) в m-й подобласти в матричном виде

$$Du = g, \tag{11}$$

где D — квадратная матрица, u — искомый вектор (индексы h и m мы опускаем), g — известный вектор. Рассмотрим аналог итерационного неявного метода Писмана–Речфорда для трехмерного случая. При решении системы (11) он может быть представлен как

$$\begin{aligned} u^{n-1/2} &= u^{n-1} - \omega_z\big(D_{xy}u^{n-1/2} + D_zu^{n-1} - g\big), \\ u^{n} \quad &= u^{n-1/2} - \omega_z\big(D_{xy}u^{n-1/2} + D_zu^{n} - g\big), \end{aligned} \tag{12}$$

где $n = 1, 2, \ldots$ — номера итераций, ω_z — итерационный числовой параметр, а D_{xy} и D_z — пятидиагональная и трехдиагональная матрицы, такие, что $D = D_{xy} + D_z$ (подробнее см. [7]). Отсюда видно, что на полуцелом шаге решаются "двумерные" системы, а на целом — "одномерные". Решение двумерных систем также проводится методом Писмана–Речфорда вида (12), в котором матрица D_{xy} представляется в виде $D_{xy} = D_x + D_y$, где D_x, D_y — трехдиагональные матрицы. Решение всех "одномерных" систем, к которым приводит реализация алгоритма вида (12), осуществляется методом прогонки.

3. Технологии распараллеливания. При распараллеливании рассматриваемой задачи применяется две парадигмы — MPI-распараллеливание и CUDA-распараллеливание, реализуемые соответственно на вычислительном кластере и вычислительных ускорителях GPU.

3.1. MPI-распараллеливание. MPI-распараллеливанию, в первую очередь, подлежит итерационный процесс по подобластям, который занимает подавляющую часть времени решения всей задачи, и, во вторую очередь, внутренний итерационный процесс (10) поиска решений на ребрах и в макроузлах. Эти итерационные процессы обладают внутренним естественным параллелизмом и не требуют дополнительных вычислительных затрат. Эффективность их распараллеливания целиком и полностью зависит от технологии проведения расчетов, от организации обменов между процессорами вычислительной сети.

Важным звеном в технологической цепи решений по достижению эффективности распараллеливания является отображение сеточных данных на вычислительную сеть. Обычно принятый способ отображения "одна подобласть–один процессор" в данном случае не эффективен, так как подобласти могут содержать различное число узлов, в которых вычисляются значения искомой функции (в дальнейшем — счетных узлов), что приводит к разбалансировке загрузки процессоров. Поэтому подобласти группируются в объединения, содержащие приблизительно одинаковое число счетных узлов, и устанавливается отображение "одно объединение–один процессор". Передача информации между подобластями одного объединения обходится без межпроцессорных обменов, а между подобластями различных объединений требуется их проведение. Инициализация и осуществление межпроцессорных обменов являются самыми медленными операциями процесса расчета. В связи с этим в основу технологии MPI-распараллеливания кладутся асинхронные операции, позволяющие хотя бы частично проводить обмены на фоне выполнения необходимых арифметических и логических операций.

3.2. CUDA-распараллеливание. Решение подзадач в каждой подобласти осуществляется при помощи трехмерного аналога метода Писмана–Речфорда, который распараллеливается на вычислительных ускорителях GPU.

CUDA — это архитектура параллельных вычислений от NVIDIA [8], позволяющая существенно увеличить вычислительную производительность благодаря использованию графических процессоров GPU (Graphics Processing Unit). GPU состоит из процессорных ядер, которые в терминологии NVIDIA называются Streaming Multiprocessor (SM), каждое из которых выполняет сотни программных нитей. Выполнение нитей на GPU происходит блоками, каждый из которых может состоять от 1 до 1024 нитей. Объединение исполняемых блоков называется гридом (Grid). В CUDA каждый блок можно представить в виде одно-, двух- или трехмерного массива, где индекс массива — это индекс исполняемой нити. Множество блоков в гриде представляется в виде одного двумерного массива.

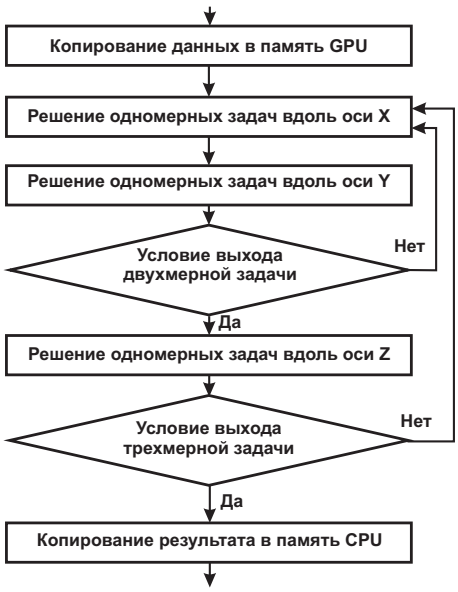

Рис. 1. Блок-схема итерационного процесса

Программу, реализующую трехмерный аналог метода Писмана–Речфорда, можно условно разделить на три части:

1) выделение памяти на GPU и копирование данных из памяти CPU;

2) запуск итерационного процесса на GPU;

3) копирование результата в память CPU и удаление выделенной памяти на GPU.

Копирование и выделение данных на GPU осуществляется средствами CUDA. Данные, являющиеся константами в итерационном процессе, копируются в константную память устройства до проведения итераций. Блок-схема проведения итерационного процесса показана на рис. 1.

Рассматриваемый алгоритм реализован в виде функции языка C. На ее вход подается информация о краевых условиях и правой части исходной краевой задачи, параметры подсетки, на которой решается рассматриваемая в данной подобласти краевая подзадача, а также счетные параметры.

Работу этой функции можно условно разделить на две части: 1) подготовка и 2) расчет. В первой части вычисляются коэффициенты для метода прогонки; выделяется память на GPU, в которую копируются коэффициенты прогонки, краевые значения и значения правых частей уравнения Пуассона. Во второй части реализован двухуровневый итерационный процесс в виде двухуровневого цикла. Во вложенном цикле на GPU запускаются два расчета одномерных задач вдоль осей OX и OY и расчет невязки двумерной задачи, по окончании которых проверяется условие на выход из цикла. В основном цикле на GPU запускается расчет одномерных задач вдоль оси OZ и расчет невязки трехмерной задачи.

4. Численные эксперименты. Цель проводимых численных экспериментов — исследование эффективности применения графических ускорителей при распараллеливании решения трехмерных краевых задач на квазиструктурированных сетках. Критерием эффективности служила величина отношения времени расчетов с использованием только CPU ко времени расчетов с использованием CPU и GPU.

Рассматривалась следующая модельная задача $\Delta u = 0$, $u\big|_\Gamma = 1$ в единичном кубе $\overline{R}$. Проводилась декомпозиция расчетной области при помощи равномерной макросетки

$$\overline{\Omega}_H = \left\{X_I = IH_x,\, Y_J = JH_y,\, Z_K = KH_z,\, I = \overline{0, N_x},\, J = \overline{0, N_y},\, K = \overline{0, N_z}\right\}$$

с шагами $H_x = \dfrac{1}{N_x}$, $H_y = \dfrac{1}{N_y}$, $H_z = \dfrac{1}{N_z}$, где N_x, N_y, N_z — заданные целые числа.

В замкнутых макроэлементах $\overline{R}_m = \overline{R}_{I,J,K}$ строились равномерные параллелепипедальные подсетки

$$\overline{\Omega}_{h,m} = \{x_{i_m} = X_I + i_m h_{x,m},\, y_{j_m} = Y_J + j_m h_{y,m},\, z_{k_m} = Z_K + k_m h_{z,m}\}$$

с шагами $h_{x,m} = \dfrac{X_{I+1} - X_I}{n_{x,m}}$, $h_{y,m} = \dfrac{Y_{J+1} - Y_J}{n_{y,m}}$, $h_{z,m} = \dfrac{Z_{K+1} - Z_K}{n_{z,m}}$, где $i_m = \overline{0, n_{x,m}}$, $j_m = \overline{0, n_{y,m}}$, $k_m = \overline{0, n_{z,m}}$.

В качестве оператора Δ_h выбирались обычные семиточечные разностные операторы Лапласа. Итерации по подобластям проводились при помощи метода сопряженных градиентов, легко поддающегося распараллеливанию.

Вычисления проводились на кластере НКС-30Т Сибирского суперкомпьютерного центра (ССКЦ СО РАН, г. Новосибирск), который имеет следующие характеристики одного вычислительного узла:

— CPU: 2 CPU Xeon X5670 6 x 2.93 GHz,

— GPU: 3 NVIDIA Tesla M 2090 6Gb на архитектуре Fermi 512 CUDA Cores.

Проведенные серии экспериментов тематически можно представить в виде трех разделов, приведенных ниже.

Ускорение в зависимости от количества ядер CPU

Ядра CPU	Сетка				
	16^3	32^3	48^3	64^3	128^3
1	5	20	39	33	45
3	4	19	37	34	47
6	4	18	37	34	47
12	4	19	38	36	47

4.1. Исследование эффективности в зависимости от числа ядер CPU. В таблице приведено отношение времени, затраченного на вычисление решения только на CPU ядрах к времени, затраченному на вычисление решения с использованием GPU. Макросетка имела параметры N_x, $N_y = 4$, $N_z = 3$, а параметры согласованных подсеток приведены в таблице. При запуске расчета на GPU-устройстве размер блока брался равным 8×8, что позволило обеспечить эффективное использование параметров GPU.

Из полученных результатов можно сделать следующие выводы:

1) ускорение, полученное с использованием GPU, с точностью до погрешностей измерения не зависит от количества ядер CPU;

2) рост ускорения с ростом числа узлов в сетке связан с возрастающим объемом вычислений, приходящихся на одну нить GPU-устройства.

4.2. Исследование эффективности на согласованных и несогласованных квазиструктурированных сетках. В проводимых экспериментах макросетка имела параметры N_x, $N_y = 4$, $N_z = 3$, а размеры подсеток по одному направлению приведены на рис. 2. Несогласованная сетка была такой, что любые две соседние подсетки имели различное, отличающееся в два раза по одному направлению, число узлов. На рис. 2 дано число узлов густых подсеток в несогласованных сетках.

Из приведенного рисунка следует, что эффективность применения GPU незначительно отличается для согласованных и несогласованных сеток. Количество блоков, вычисляемое на GPU, зависит от размера сетки. Если это количество не кратно количеству блоков, которое GPU-устройство может вычислять одновременно, то это ведет к снижению эффективности вычисления. Данное свойство объясняет нелинейность ускорения, изображенного на рис. 2.

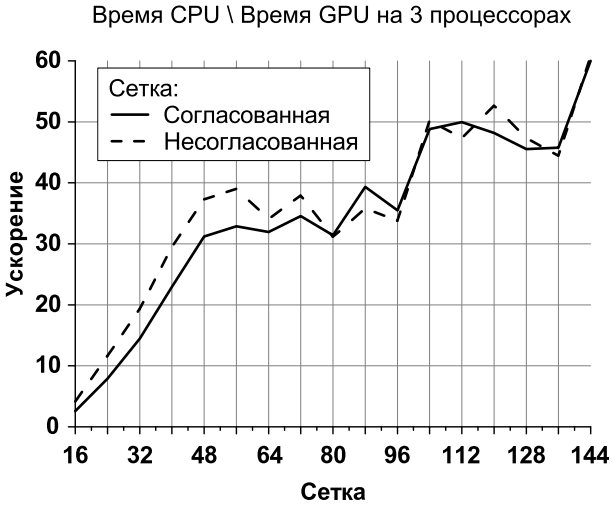

Рис. 2. Ускорение, полученное на различных сетках на 3 ядрах CPU

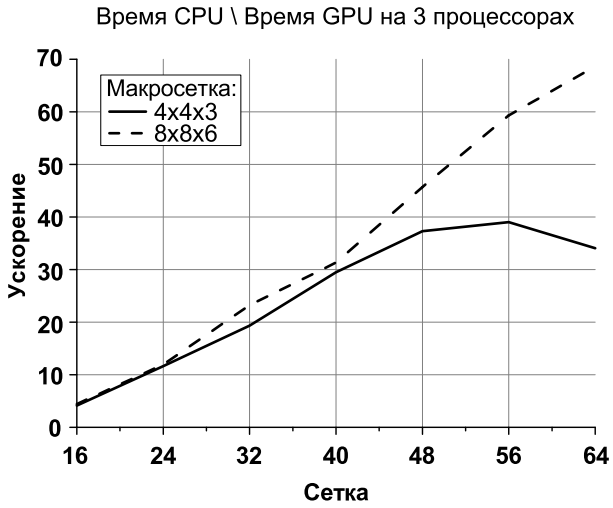

Рис. 3. Ускорение при разном количестве узлов макросетки

4.3. Исследование эффективности в зависимости от размеров макросетки. На рис. 3 показана зависимость полученного ускорения в зависимости от размера сетки в подобласти при различных размерах макросетки. Различие в ускорениях объясняется тем, что с ростом количества узлов макросетки растет время на обмен и обработку данных между подобластями на одном ядре CPU.

5. Заключение. В настоящей статье исследована эффективность применения графических ускорителей при распараллеливании решения трехмерных краевых задач на квазиструктурированных сетках. Распараллеливание осуществляется методом декомпозиции расчетной области на подобласти, сопрягаемые без наложения, основанном на прямой конечно-разностной аппроксимации уравнения Пуанкаре–Стеклова на интерфейсе. Возникающий при этом итерационный процесс по подобластям распараллеливается на CPU. Для выполнения процедуры решения подзадач в подобластях, отнимающей наибольшую часть времени решения всей задачи, применяется метод Писмана–Речфорда, который обладает быстрой сходимостью. Его распараллеливание осуществляется на GPU в системе CUDA. Показано, что применение графических ускорителей значительно (более 60 раз) сокращает время решения задачи по сравнению с расчетами только на CPU.

Работа выполнена при финансовой поддержке Российского научного фонда (проект № 14–11–00485) и РФФИ (проект № 16–01–00168).

Статья рекомендована к публикации Программным комитетом Международной научной конференции “Параллельные вычислительные технологии” (ПаВТ–2016; http://agora.guru.ru/pavt2016).

СПИСОК ЛИТЕРАТУРЫ

1. *Василевский Ю.В., Ольшанский М.А.* Краткий курс по многосеточным методам и методам декомпозиции области. М.: Изд-во Моск. ун-та, 2007.
2. *Quarteroni A., Valli A.* Domain decomposition methods for partial differential equations. Oxford: Clarendon Press, 1999.
3. *Свешников В.М.* Построение прямых и итерационных методов декомпозиции // Сибирский журнал индустриальной математики. 2009. **12**, № 3. 99–109.
4. *Свешников В.М., Беляев Д.О.* Построение квазиструктурированных локально-модифицированных сеток для решения задач сильноточной электроники // Вестник ЮУрГУ. Серия матем. моделирование и программирование. 2012. № 40. 130–140.
5. *Свешников В.М., Рыбдылов Б.Д.* О распараллеливании решения краевых задач на квазиструктурированных сетках // Вестник ЮУрГУ. Серия вычислит. матем. и информатика. 2013. **2**, № 3. 63–72.
6. *Корнеев В.Д.* Параллельное программирование в MPI. Новосибирск: Ин-т вычислит. матем. и матем. геофиз., 2002.
7. *Ильин В.П.* Методы конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск: Ин-т вычислит. матем. и матем. геофиз., 2000.
8. NVIDIA. CUDA C Best Practices Guide. http://docs.nvidia.com/cuda/cuda-c-best-practices-guide.

Поступила в редакцию
10.01.2016

Parallelization Technologies for Solving Three-Dimensional Boundary Value Problems on Quasi-Structured Grids Using the CPU+GPU Hybrid Computing Environment

I. A. Klimonov [1], V. D. Korneev [2], and V. M. Sveshnikov [3]

[1] *Novosibirsk State University, Faculty of Mechanics and Mathematics; ulitsa Pirogova 2, Novosibirsk, 630090, Russia; Student, e-mail: ilya.klimonov@gmail.com*

[2] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Ph.D., Associate Professor, e-mail: korneev@ssd.sscc.ru*

[3] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Dr. Sci., Professor, Head of Laboratory, e-mail: victor@lapasrv.sscc.ru*

Received January 10, 2016

Abstract: When parallelizing the solution processes of solving three-dimensional boundary value problems on quasi-structured grids by the method of decomposition of the computational domain into subdomains without imposition, the most time consuming computational procedure is a solution of subproblems in subdomains. The application of parallelepiped quasi-structured grids makes it possible to use the rapidly convergent method of alternating directions. The parallelization of iterative processes on subdomains is performed on CPU using MPI. In order to solve the subproblems, we propose to use the graphics accelerators (GPU). Experimental results of using the graphics accelerators to solve the subproblems by Peaceman–Rachford method are discussed. The computational acceleration achieved on the CPU+GPU hybrid computing environment is experimentally estimated compared to using the CPU only.

Keywords: boundary value problems, domain decomposition methods, Poincaré–Steklov equation, quasi-structured grids, Peaceman–Rachford method, graphics accelerators.

References

1. Yu. V. Vasilevskii and M. A. Ol'shanskii, *A Short Course on Multigrid and Domain Decomposition Methods* (Mosk. Gos. Univ., Moscow, 2007) [in Russian].

2. A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations* (Clarendon Press, Oxford, 1999).

3. V. M. Sveshnikov, "Construction of Direct and Iterative Decomposition Methods," Sib. Zh. Ind. Mat. **12** (3), 99–109 (2009) [J. Appl. Ind. Math. **4** (3), 431–440 (2010)].

4. V. M. Sveshnikov and D. O. Belyaev, "Construction of Quasi-Structured Locally Modified Grids to Solve the Problems of High Current Electronics," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Mat. Model. Programm., No. 40, 130–140 (2012).

5. V. M. Sveshnikov and B. D. Rybdylov, "About Parallelization of Solving of Boundary Value Problems on Quasistructured Grids," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **2** (3), 63–72 (2013).

6. V. D. Korneev, *Parallel Programming with MPI* (Inst. Comput. Math. Math. Geophys., Novosibirsk, 2002) [in Russian].

7. V. P. Il'in, *Finite Difference and Finite Volume Methods for Elliptic Equations* (Inst. Comput. Math. Math. Geophys., Novosibirsk, 2000) [in Russian].

8. NVIDIA. CUDA C Best Practices Guide. http://docs.nvidia.com/cuda/cuda-c-best-practices-guide. Cited February 22, 2016.