Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного программирования

Белов Никита Андреевич, 627 группа

Решение задачи Дирихле для уравнения Пуассона методом конечных разностей

Отчет по второму заданию по курсу «Суперкомпьютерное моделирование и технологии»

Москва, 2016

Содержание

1 Математическая постановка задачи 3

2 Численный метод решения задачи 4

3 Описание программной реализации 5

3.1 OpenMP 6

3.2 Сохранение и загрузка состояния 6

4 Результаты расчетов 7

4.1 IBM Blue Gene/P 7

4.2 «Ломоносов» 8

5 Графическое изображение решений 10

Список литературы 11

1 Математическая постановка задачи

Вариант 2: набор данных 1, равномерная сетка, максимум-норма.
В прямоугольной области

$$\Pi = [0, 3] \times [0, 3] \tag{1}$$

необходимо найти дважды гладкую функцию

$$u = u(x, y), \tag{2}$$

удовлетворяющую дифференциальному уравнению:

$$-\triangle u = \frac{x^2 + y^2}{(1 + xy)^2}, \tag{3}$$

где

$$x \in [0, 3], y \in [0, 3] \tag{4}$$

и

$$\triangle u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{5}$$

а также удовлетворяющую дополнительному условию

$$u(x, y) = ln(1 + xy) \tag{6}$$

во всех граничных точках (x, y) прямоугольника.

2 Численный метод решения задачи

Вариант 2: набор данных 1, равномерная сетка, максимум-норма.

Для решения задачи был использован метод *скорейшего спуска* (на первой итерации) и метод *сопряженных градиентов* (на последующих итерациях) на равномерной сетке при заданном количестве точек N_x и N_y по осям Ox и Oy соответственно:

$$x_i = 3\frac{i}{N_x}, y_j = 3\frac{j}{N_y}. \tag{7}$$

Полностью численный метод решения задачи описан в [1].

Опишем условие остановки итерационного процесса. Пусть $P^{(n)} = [p_{ij}^{(n)}]$ – приближенное решение, полученное на итерации n. Максимум-норма:

$$\| p \| = max_{\substack{0<i<N_1 \\ 0<j<N_2}} |p_{ij}|, \tag{8}$$

где $P = [p_{ij}]$.

Итерационный процесс останавливается, как только:

$$\| P^{(n)} - P^{(n-1)} \| < \epsilon, \tag{9}$$

где $\epsilon = 0.0001$.

3 Описание программной реализации

Программная реализация была выполнена на языке C с использованием стандарта `C11`. Для распараллеливания решения задачи на разных вычислительных узлах использовалась библиотека `MPI`. Для распараллеливания в рамках одного вычислительного узла (содержащего несколько вычислительных ядер) была использована технология OpenMP. Исходный код содержит собственную библиотеку для более удобной работы с динамическими массивами в языке C `array.h`.

В разработанной программе перед решением поставленной задачи инициализируется декартова топология, разбивается прямоугольник Π и инициализируется сетка и используемые массивы.

После чего начинается итерационный процесс, останавливающийся при выполнении ранее указанного неравенства (9). На каждой итерации каждый процесс получает и отправляет своим соседям свои граничные строки и столбцы (функция `neighbors_exchange`), а затем использует эти значения. Происходит следующее число обменов:

1) **0**, если процесс граничный в топологии,

2) **2**, если процесс угловой,

3) **3**, если процесс боковой,

4) **4**, если процесс внутренний.

При вычислении коэффициентов α и τ каждый процесс получает и суммирует вычисленные значения со значениями с других процессов.

После окончания итерации процессы вычисляют значение нормы по указанной в предыдущем разделе формуле, затем обмениваются вычисленными значениями и каждый процесс хранит максимальное, после чего проверяется критерий завершения итерационного процесса (9).

Затем в процессе с ранком 0 происходит сбор решения от всех процессов в единый массив (функция `make_solution`).

3.1 OpenMP

По условиям задания было необходимо реализовать версию программы с использованием `OpenMP`. Это было реализовано следующим образом.

Ограничение процессорных ядер до трех вместо четырех было сделано с использованием функции `omp_set_num_threads(3);` в начале функции `main`. Затем для каждого цикла, выполняющего операции над векторами, была использована директива `#pragma omp parallel for`, которая распределяет итерации между параллельными потоками. Для каждого цикла, выполняющего операции над переменной `varname` (например, вычисление суммы), была использована директива `#pragma omp parallel for reduction(+:varname)`, которая также распределяет итерации между параллельными потоками, но при этом хранит свою локальную копию переменной, после чего все локальные копии сливаются в исходную переменную.

3.2 Сохранение и загрузка состояния

Так как для программ студентов время расчетов на ПВС «IBM Blue Gene/P» сильно ограничено (300 секунд), то был написан механизм сохранения и загрузки состояния расчетов.

К аргументам коммандной строки программы добавлен третий параметр, который может принимать следующие значения:

1) **save** – при использовании этого параметра программа сохранит расчеты через число итераций, указанное в четвертом параметре;

2) **loadsave** – при использовании этого параметра программа загрузит сохраненные ранее расчеты, а затем сохранит новые расчеты через число итераций, указанное в четвертом параметре;

3) **load** – при использовании этого параметра программа загрузит сохраненные ранее расчеты.

4 Результаты расчетов

4.1 IBM Blue Gene/P

Были проведены расчеты для следующего числа процессоров: 1, 128, 256, 512 на сетках с числом узлов 1000×1000 и 2000×2000 без использования `OpenMP` и с использованием (гибридная программа). Результаты расчетов представлены в таблицах 1 и 2 соответственно. На рисунках 1 и 2 представлены графики зависимости ускорения от числа процессоров для решений MPI и MPI+OpenMP для сеток 1000×1000 и 2000×2000 точек соответственно.

Число процессоров N_p	Число точек сетки N^2	Время решения T, сек	Ускорение S
1	1000×1000	1652.024522	
128	1000×1000	12.466822	132.514
256	1000×1000	4.089699	403.948
512	1000×1000	3.112727	530.732
1	2000×2000	13587.828157	
128	2000×2000	107.145814	126.816
256	2000×2000	54.971916	247.178
512	2000×2000	23.605630	575.618

Таблица 1. Таблица с результатами расчетов на ПВС «IBM Blue Gene/P» без использования `OpenMP`.

Число процессоров N_p	Число точек сетки N^2	Время решения T, сек	Ускорение S
1	1000×1000	440.038644	
128	1000×1000	3.567329	123.352
256	1000×1000	1.373013	320.491
512	1000×1000	1.308485	336.296
1	2000×2000	2932.124473	
128	2000×2000	24.786318	118.296
256	2000×2000	13.203982	222.064
512	2000×2000	7.093203	413.371

Таблица 2. Таблица с результатами расчетов на ПВС «IBM Blue Gene/P» с использованием `OpenMP`.

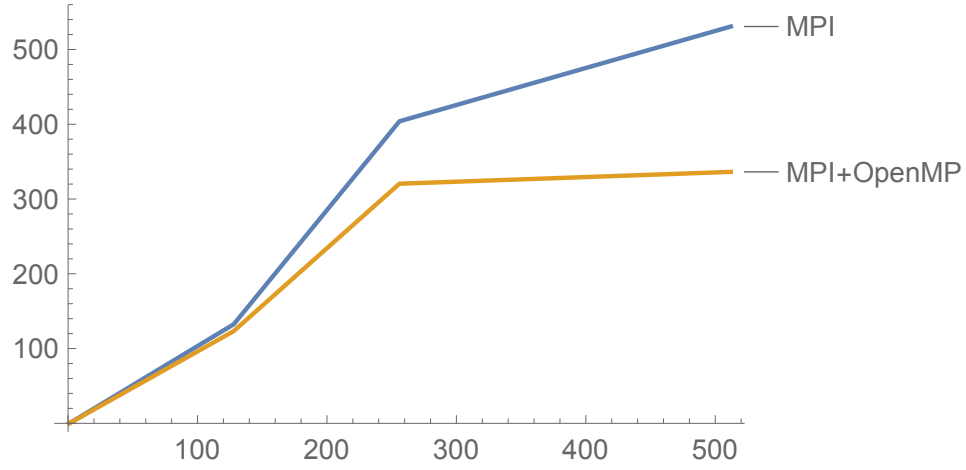

Рисунок 1. График зависимости ускорения от числа процессоров для решений MPI и MPI+OpenMP для сетки 1000×1000 точек.

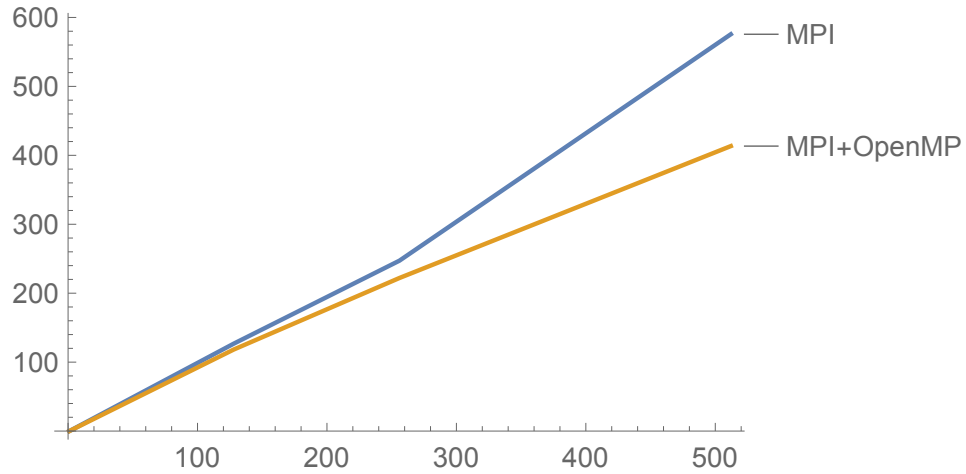

Рисунок 2. График зависимости ускорения от числа процессоров для решений MPI и MPI+OpenMP для сетки 2000×2000 точек.

4.2 «Ломоносов»

Были проведены расчеты для следующего числа процессоров: 1, 8, 16, 32, 64, 128 на сетках с числом узлов 1000×1000 и 2000×2000. Результаты представлены в таблице 3.

Число процессоров N_p	Число точек сетки N^2	Время решения T, сек	Ускорение S
1	1000 × 1000	234.741683	
8	1000 × 1000	29.986969	7.828
16	1000 × 1000	15.227095	15.416
32	1000 × 1000	7.721868	30.400
64	1000 × 1000	4.138745	56.718
128	1000 × 1000	2.141562	109.612
1	2000 × 2000	2123.040323	
8	2000 × 2000	238.554619	8.900
16	2000 × 2000	120.629342	17.600
32	2000 × 2000	60.860912	34.884
64	2000 × 2000	30.859959	68.796
128	2000 × 2000	15.950209	133.104

Таблица 3. Таблица с результатами расчетов на ПВС «Ломоносов».

5 Графическое изображение решений

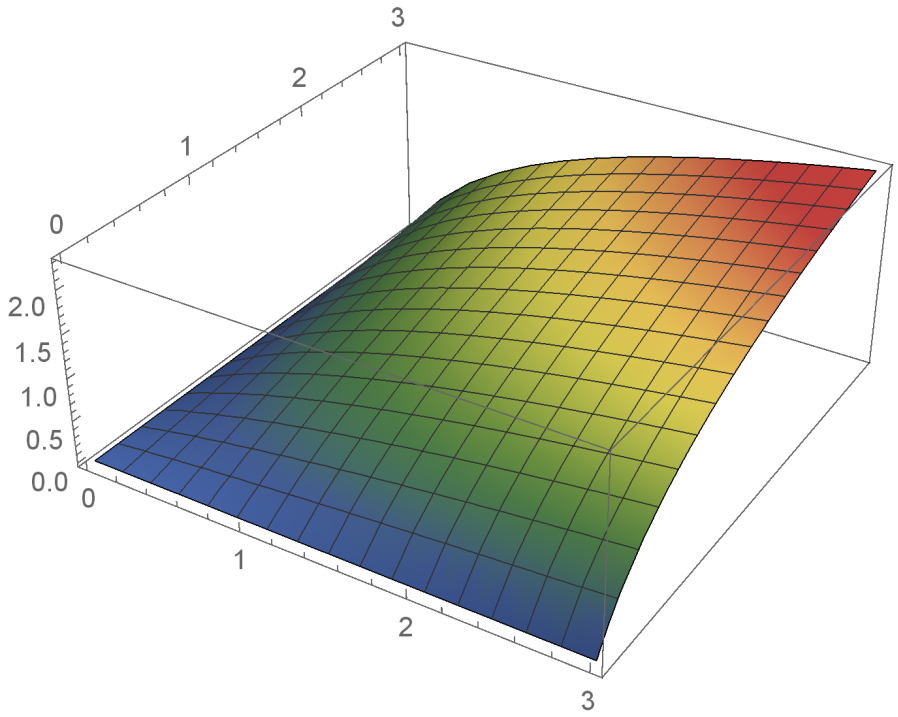

Рисунок 3. Точное решение.

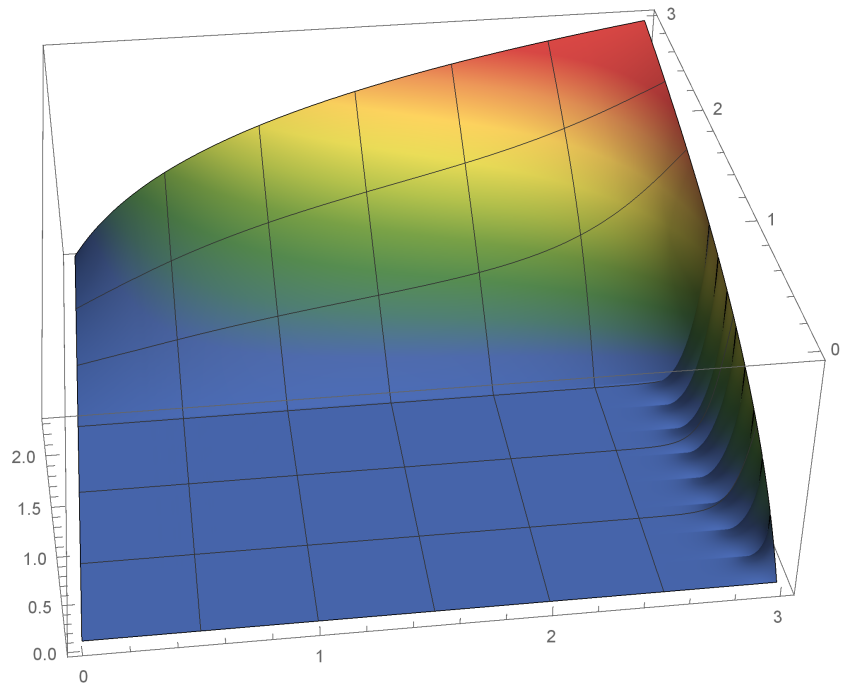

Рисунок 4. Приближенное решение, полученное на сетке 2000×2000 точек.

Список литературы

1. СКИ. Задание по курсу «Суперкомпьютерные моделирование и технологии». МГУ ВМК, Москва, Россия: Октябрь, 2016. 8 с.