МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №1
по курсу «Параллельная обработка данных»

Message Passing Interface

Выполнил: А. Е. Максимов
Группа: М8О-407Б-19
Преподаватель: А. Ю. Морозов

Москва, 2023

1 Условие

Цель работы: Знакомство с технологией MPI. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в двухмерной области с граничными условиями первого рода.

Вариант на "два". Решение двумерной задачи.

Требуется реализовать решение статической двухмерной задачи Дирихле для уравнения Лапласа с граничными условиями первого рода.

В реализации необходимо использовать следующие методы из библиотек MPI:

- обмен граничными слоями с помощью функции `MPI_Bsend`;
- контроль сходимости с помощью функции `MPI_Allreduce`;

2 Программное и аппаратное обеспечение

Характеристики системы:

- Процессор: «Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz 6 ядер 12 потоков»
- Память: 16,0 ГБ встроенной памяти ноутбука
- SSD: «Hynix BC511 NVMe SK hynix 512GB »

Программное обеспечение:

- ОС: «Windows 10 Домашняя x64»
- Среда разработки: «Microsoft Visual Studio Communuty edition»
- Компилятор: «Microsoft® C-C++ Compiler Driver 19.35.32216.1»

3 Метод решения

Алгоритм вычисления

Для вычисления результатов использовалась конечно-разностная сетка ω_{h_x,h_y} над двумерными координатами $x\epsilon[0,l_x]; y\epsilon[0,l_y]$

$$\omega^{\tau}_{h_x,h_y} = \{x_i = i * h_x; i\epsilon[1,n_x]; y_j = j * h_y; j\epsilon[1,n_y]\}$$

Для вычисления распределения температур мы будем использовать временные слои, которые будем отражать следующим видом: $u^{(0)}, ..., u^{(k)}, u^{(k+1)}, ...$

Изначальное значение слоя $u^{(0)}$ вводится при запуске программы, также как и граничные значения $u_{left}, u_{right}, u_{down}, u_{up}$.

Новый временной слой вычисляется по формуле:

$$u^{(k+1)}_{i,j} = \frac{(u^{(k)}_{i+1,j} + u^{(k)}_{i-1,j})h_x^{-2} + (u^{(k)}_{i,j+1} + u^{(k)}_{i,j-1})h_y^{-2}}{2(h_x^{-2} + h_y^{-2})}$$

где

$$i = 1..n_x, j = 1..n_y;$$
$$h_x = l_x n_x^{-1}, h_y = l_y n_y^{-1};$$
$$u^{(k)}_{o,j} = u_{left}, u^{(k)}_{n_x+1,j} = u_{right};$$
$$u^{(k)}_{i,0} = u_{down}, u^{(k)}_{i,n_y+1} = u_{up};$$
$$u^{(0)}_{i,j} = u^{(0)}$$

Итерационный процесс завершается, когда

$$\max_{i,j} |u^{(k+1)}_{i,j} - u^{(k)}_{i,j}| < \varepsilon$$

Реализация

Для организации параллельного вычисления новых значений $u_{i,j}^{(k+1)}$, вся конечно-разностная сетка разделяется на равные блоки, размеры `bl_sz_x` и `bl_sz_y` которых подаются в поток ввода при запуске программы. Сами блоки запускаются в координатной сетке процессов, размерность которой `bl_cnt_x` и `bl_cnt_y` также считывается программой.

В начале вычисления каждого нового слоя все блоки обмениваются граничными значениями через процедуры `MPI_Bsend` и `MPI_Recv`. В случае, если блок находится на границе сетки процессов, он заполняет скои граничные значения соответствующими значениями из списка $u_{left}, u_{right}, u_{down}, u_{up}$.

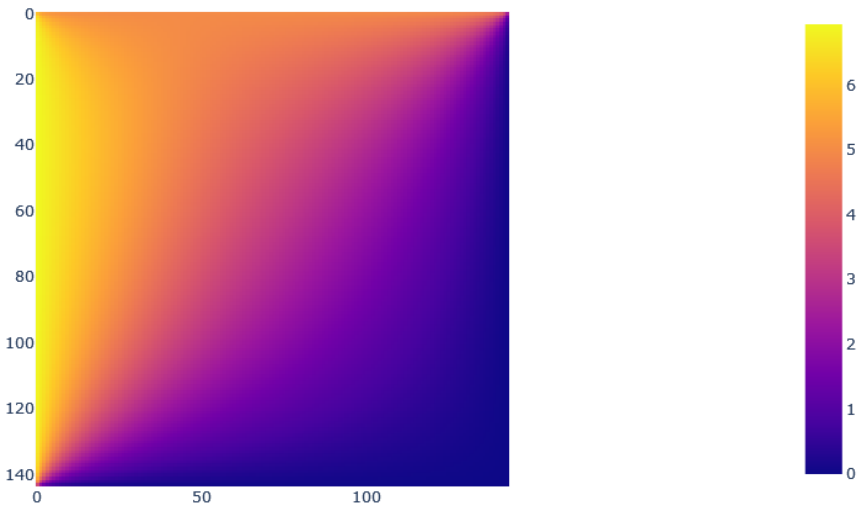
Для вычисления условий завершения алгоритма все процессы обмениваются значениями $\max_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}|$ с помощью процедуры `MPI_Allreduce`, что позволяет на уровне вызова одной функции вычислить максимальное значение среди всех блоков.

4 Результаты

Замеры времени работы CPU с разным количеством процессов, сетка имеет размер nxn

Кол-во процессов	$n = 12$, мс	$n = 24$, мс	$n = 48$, мс	$n = 96$, мс	$n = 144$, мс
1, без MPI	0.998	15.052	197.104	2767.28	13379.2
1	1.268	16.085	221.380	3085.717	15353.774
2	1.985	13.201	136.050	1775.00	8155.761
3	3.199	15.040	121.11	1348.411	6137.803
4	3.258	18.054	113.024	1142.815	5133.109
6	7.195	25.130	123.332	1086.646	4479.348
8	6.943	24.450	122.554	972.204	3932.533
12	10.139	42.664	150.231	1035.094	3543.896
16	16554.404	58625.509	205469.292	$>$ 320 сек	$>$ 320 сек

Визуализация результатов



Запуск производился с параметрами

`bl_sz_x` $= 48$ и `bl_sz_y` $= 72$ `bl_cnt_x = 3` и `bl_cnt_y = 2`

Значения границ и начального слоя были введены следующие:

$l_x = 1.0; l_y = 1.0; u_{left} = 7.0, u_{right} = 0.0, u_{down} = 5.0, u_{up} 0.0, u^{(0)} 0.0$

5 Выводы

Решение задачи Дирихле имитирует двухмерную задачу теплопроводности. Конечно-разностный метод численного решения дифференциальных уравнений является самым распространённым и используемым.

Метод Якоби позволяет итерационным процессом приблизиться к решению с заданой точностью, обеспечивая скорость схождения $O(k * n_x * n_y)$, где n_x - размер сетки по оси абсцисс, n_y- размер сетки по оси ординат, k - количество итераций. Это не самый быстрый метод, но он демонстрирует основы конечно-разностного подхода к решению подобных задач.

В рамках исследования времени работы были получены данные, согласно которым наиболее оптимальное количество параллельных потоков - от шести до двенадцати, что совпадает с количеством ядер/потоков на моём процессоре. При превышении числа потоков время выполнения многократно увеличилось, поскольку часть процессов находилась в режиме ожидания, что только ухудшалось с необходимостью взаимодействия каждого потока с четырьмя соседями.