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 (MPI).

Выполнил: И. П. Моисеенков
Группа: М8О-408Б-19
Преподаватель: А.Ю. Морозов

Москва, 2022

Условие

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

$$\frac{d^2u(x,y,z)}{dx^2} + \frac{d^2u(x,y,z)}{dy^2} + \frac{d^2u(x,y,z)}{dz^2} = 0,$$

$$u(x \le 0, y, z) = u_{left},$$
$$u(x \ge l_x, y, z) = u_{right},$$
$$u(x, y \le 0, z) = u_{front},$$
$$u(x, y \ge l_y, z) = u_{back},$$
$$u(x, y, z \le 0) = u_{down},$$
$$u(x, y, z \ge l_z) = u_{up}.$$

Вариант 2. Обмен граничными слоями через bsend, контроль сходимости allgather.

Входные данные. На первой строке заданы три числа: размеры сетки процессов. Гарантируется, что при запуске программы количество процессов будет равно произведению этих трех чисел. На второй строке задается размер блока, который будет обрабатываться одним процессом: три числа. Далее задается путь к выходному файлу, в который надо записать конечный результат работы программы, и точность. На последующих строках описывается задача: задаются размеры области, граничные условия и начальное значение u.

Выходные данные. В файл, определенный во входных данных, необходимо напечатать построчно значения в ячейках сетки в формате с плавающей запятой с семью знаками мантиссы.

Метод решения

Создаем трехмерную сетку. С каждой ячейкой сопоставляется значение функции u в точке, соответствующей центру ячейки. Граничные условия реализуются через виртуальные ячейки, которые окружают рассматриваемую область.

Каждый процесс будет обрабатывать свой кусок сетки. Поиск решения сводится к итерационному процессу:

$$u_{i,j,k}^{(n+1)} = \frac{\left(u_{i+1,j,k}^{(n)} + u_{i-1,j,k}^{(n)}\right)h_x^{-2} + \left(u_{i,j+1,k}^{(n)} + u_{i,j-1,k}^{(n)}\right)h_y^{-2} + \left(u_{i,j,k+1}^{(n)} + u_{i,j,k-1}^{(n)}\right)h_z^{-2}}{2\left(h_x^{-2} + h_y^{-2} + h_z^{-2}\right)},$$

Процесс останавливается, когда изменение значений функции после некоторой итерации стало меньше заданного эпсилон.

Описание программы

Будем параллелить вычисления на сетке - у каждого процесса будет свой кусок сетки.

Сам алгоритм будет состоять из трех шагов:

1. Обмен граничными условиями между процессами. После каждой итерации процессы должны узнавать новое значение на своих границах. Поэтому будем сообщать эту информацию процессам с помощью bsend.
2. Вычисление обновленного значения функции (по формуле выше).
3. Считаем погрешность в каждом процессе и выбираем максимальную. Сравниваем ее с эпсилоном. В техническом плане это реализовано через отправку погрешностей всеми процессами в главный процесс с помощью allgather.

Итоговый ответ будем также отправлять в главный процесс. Для этого можно использовать send/recv.

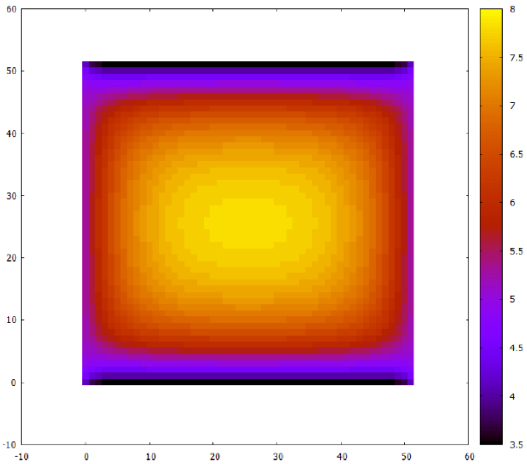
Результаты

Рассмотрим время работы программы с разным количеством процессов. Тест будет одинаковый, размер сетки - 40x40. Различаться будет только количество блоков (и соответственно количество процессов). Время считывания данных и печати результата не учитываем. Результаты приведены в таблице ниже.

Количество процессов	Время работы, мс
1	19,22
2	9,98
4	7,29
8	9,99
16	17,24

Если количество процессов становится больше количества ядер процессора, то общее время работы программы увеличивается. В данном случае наиболее оптимальное решение - использовать 4 процесса при распараллеливании.

Посмотрим на получившийся результат. Для этого зафиксируем какой-нибудь z (я для примера возьму z = 10). Посмотрим на значения температуры на плоскости.



Выводы

В данной лабораторной работе я познакомился с технологией MPI для параллельной обработки данных. С помощью MPI мы можем создать несколько процессов, которые будут работать параллельно. При этом они могут довольно просто

обмениваться друг с другом информацией. Поэтому MPI удобен при решении сложных в точки зрения вычислений математических задач.