МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу «Параллельная обработка данных»

Технология MPI и технология OpenMP
Вариант 3. Распараллеливание в общем виде с разделением работы между нитями вручную (“в стиле CUDA”).

Выполнил: А.В. Синявский
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы.

Совместное использование технологии MPI и технологии
OpenMP. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в трехмерной области с граничными условиями первого рода.

Входные данные.

На первой строке заданы три числа: размер 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 необходимо записать конечный результат работы программы и точность ε. На последующих строках описывается задача: задаются размеры области l_x, l_y и l_z, граничные условия: u_{down}, u_{up}, u_{left}, u_{right}, u_{front}, u_{back}, и начальное значение u^0.

Выходные данные.

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

Программное и аппаратное обеспечение

Nvidia GeForce GTX 660

Compute capability: 3.0
Графическая память: 2048MB
Регистров на блок: 65536
Нитей на блок: 1024
Мультипроцессоров: 5
Всего ядер: 960

Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz

Тактовая частота: 3.4 GHz
Кэш-память: 6 MB

Оперативная память

Объём: 8 GB

Жёсткий диск

Объём: 2 TB

Программное обеспечение

OS: Windows 10
IDE: Visual Studio Code
CUDA: v10.2 nvcc

Метод решения

Пусть нулевой процесс считывает начальные данные и отправляет всем остальным процессам. Далее каждый процесс заполняет изначальные данные и начинает вычислительный цикл. В этом цикле процессы сначала обмениваются граничными значениями (необходимо сделать несколько условий по расположению процессов в сетке, чтобы ни один из граничных процессов не повис в вечном ожидании данных), затем для каждой ячейки вычисляют новое значение, параллельно считая максимальную ошибку в рамках процесса. Затем процессы обмениваются друг с другом этой максимальной ошибкой. Максимум из максимальных ошибок по всем процессам – критерий остановки вычислительного цикла для всех процессов.
После остановки вычислительного цикла все процессы записывают результаты вычислений в один общий файл, согласно пользовательскому типу данных.

Описание программы

Практически весь код находится в функции main, так что описание придётся указывать по строчкам кода.

1. Стр.9-10
   Макросы, переводящие трёхмерную индексацию в одномерную для сетки внутри блока/сетки самих блоков соответственно.

2. Стр. 41-60
   Считывание данных и передача всем процессам

3. Стр. 70-121
   Выделение памяти и нициализация данных в рамках каждого процесса(блока)

4. Стр. 127-170
   Определение пользовательских типов данных для передачи границ между процессами

5. Стр. 181-211
   Обмен границами между соседними блоками.

6. Стр. 215-232
   Основной вычислительный цикл, распараллеленный с помощью OpenMP

7. Стр. 233-245
   Определение критерия остановки выч. цикла.

8. Стр. 249-282
   Определение пользовательского типа данных для параллельной бесконфликтной записи в файл

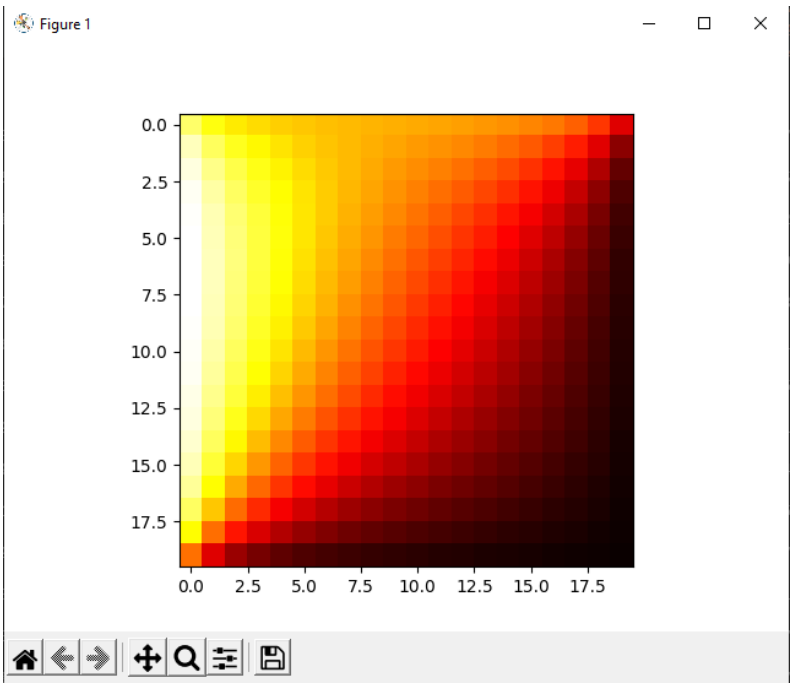
9. 284-293
   Определение параметров записи в файл для конкретного процесса и сама запись.

Результаты

Замеры времени. Общий размер сетки – 20x20x40 ячеек

Число процессов	CPU	GPU	OpenMP
1	1.667767 с	0.742268 с	1,125640 с
2	0.837142 с	0.801399 с	0,813335 с
16	0.744788 с	0.948677 с	2,491667 с
64	1.41726 с	1.035669 с	6,001248 с

Визуализация результатов.
Изображение построено по результатам теста
1 1 1
20 20 1
mpi.out
1e-10
1.0 1.0 2.0
7.0 0.0 5.0 0.0 3.0 0.0
5.0



Выводы

Проделав работу, я познакомился с технологией OpenMP, научился распараллеливать с её помощью вложенные циклы в общем виде, а также улучшил свои навыки в обращении с пользовательскими типами данных MPI.