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

Выполнил: К.О. Вахрамян
Группа: 8О-406Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

1. 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.

2. Вариант задания.

Распараллеливание основных циклов через parallel for (+директива reduction для вычисления погрешности);

Программное и аппаратное обеспечение

GPU:

--- General Information for device ---
Name: NVIDIA GeForce GTX 1650
Compute capability: 7.5
Clock rate: 1560000
Device copy overlap: Enabled
Kernel execution timeout : Enabled
--- Memory Information for device ---
Total global mem: 4100521984
Total constant Mem: 65536
Max mem pitch: 2147483647
Texture Alignment: 512
--- MP Information for device ---
Multiprocessor count: 16
Shared mem per mp: 49152
Registers per mp: 65536
Threads in warp: 32
Max threads per block: 1024
Max thread dimensions: (1024, 1024, 64)
Max grid dimensions: (2147483647, 65535, 65535)

CPU:

Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
Address sizes: 39 bits physical, 48 bits virtual
CPU(s): 8
On-line CPU(s) list: 0-7
Thread(s) per core: 2
Core(s) per socket: 4
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6

Model: 158
Model name: Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz
Stepping: 13
CPU MHz: 1274.759
CPU max MHz: 2400.0000
CPU min MHz: 800.0000
BogoMIPS: 4800.00
Virtualization: VT-x
L1d cache: 128 KiB
L1i cache: 128 KiB
L2 cache: 1 MiB
L3 cache: 8 MiB

OS:
Linux Mint 20

Compiler:
nvcc

Code Editor:
VS Code

Метод решения
Метод решения аналогичен предыдущим, однако обмен граничными значениями реализован при помощи типа MPI_Type_vector а цикл инициализации начальным условием и цикл подсчета новых значений распараллелин при помощи технологии openMP
Описание программы
Хотя мы решаем двухмерную задачу, данные хранятся в одномерном массиве. Для правильной индексации используется макрос:
```
#define _i(i, j) (((j) + 1) * (nx + 2) + (i) + 1)
```

Для индексации по сетке блоков:
```
#define _ib(i, j) ((j) * nbx + (i))
```

Данные считывает нулевой процесс и отправляет всем при помощи функции:
```
MPI_Bcast();
```

Отправка данных между процессами осуществляется при помощи буферизованной функции:
```
MPI_Bsend();
```

Данная функция является неблокирующей
Получение про помощи:

MPI_Recv();

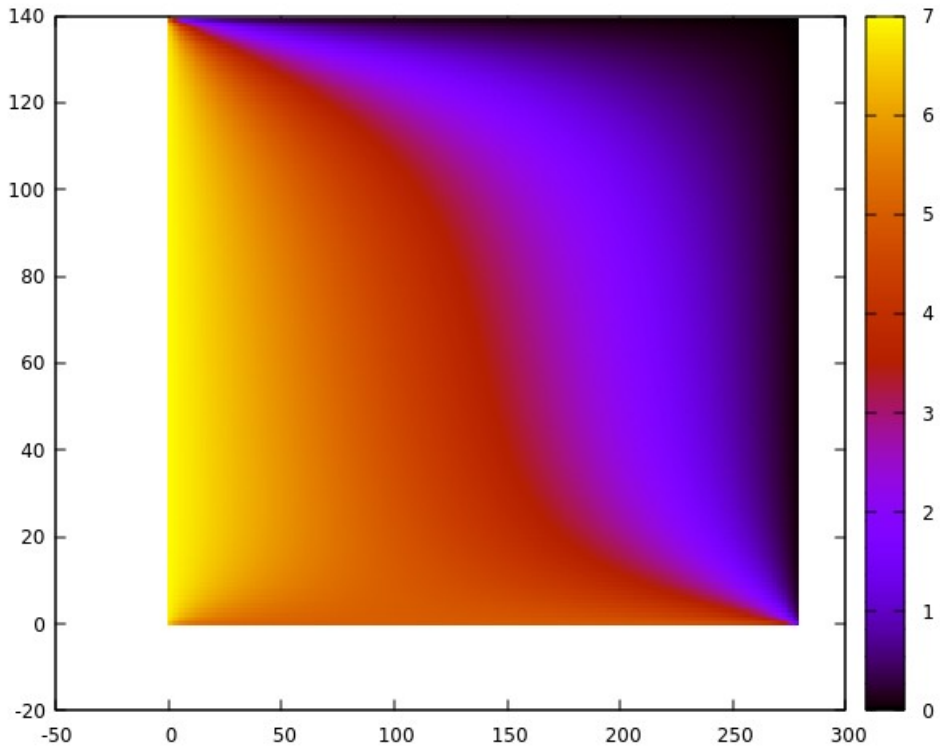
#pragma omp parallel for private(i,j) shared(data, u0)
Директива openMP для распараллеливания цикла инициализации

#pragma omp parallel for private(i,j) shared(data, next,nx,ny,hx,hy) reduction(max:check)
Директива openMP для распараллеливания цикла вычисления новых значений.

Результаты

Сетка | Блок	MPI + openMP	CPU
1 1 | 10 10	026s	0.30s
2 2 | 50 50	2.85s	3.83s

4 2
70 70
mpi.out
1e-10
1.0 2.0
7.0 0.0 5.0 0.0



Выводы

За счет использования произвольных типов данных MPI, последняя работа показалась самой простой в цикле из 3-х. Технология openMP стала интересным открытием, ведь после цикла ПГП распараллеливание ассоциируется с чем-то сложным, однако главная особенность данного стандарта — простота в использовании.