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: А. О. Дубинин
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2020

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.

Вариант 1. Распараллеливание основных циклов через parallel for (+директива reduction для вычисления погрешности);

Программное и аппаратное обеспечение

GeForce 940MX

Compute capability:	5.0
Dedicated video memory:	4096 MB
shared memory per block:	49152 bytes
constant memory:	65536 bytes
Total number of registers available per block:	65536
Maximum number of threads per multiprocessor:	2048
Maximum number of threads per block:	1024
(3) Multiprocessors, (128) CUDA Cores/MP:	384 CUDA Cores

Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz

Architecture:	x86_64
Byte Order:	Little Endian
CPU(s):	4
Thread(s) per core:	2
Core(s) per socket:	2
CPU MHz: CPU max MHz: CPU min MHz:	713.848 3100,0000 400,0000
L1d cache: L1i cache: L2 cache: L3 cache:	64 KiB 64 KiB 512 KiB 3 MiB

RAM	8GiB SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2400 MHz (0,4 ns)

SSD(SPCC_M.2_SSD)	223,6G

HDD(ST1000LM035-1RK172)	931,5G

OS: Ubuntu 20.04 focal
IDE: jetbrains clion
compiler: mpic++

Метод решения

Основная логика решения данной ЛР, была взята с лабораторной работы №7. Изменения произошли только с тем, что мы паралелилим наши циклы. Тут было важно не переусердствовать, так как при распараллеливании всех циклов большое кол-во времени тратилось на поддержку параллелизма openmp. Поэтому я оставил директиву только на перекопированнии данных после MPI_Recv и на основной цикл. Так же была добавлена редукция, которая даже при паралеливании сохраняла максимальное значение.

Описание программы

Данная лабораторная работа не сильно отличается от 7 ЛР. Это очень хорошо видно по вызову утилиты diff. Все различие заключается лишь в распараллеливании циклов for. И как мы видим, в openmp это делается очень просто одной директивой.

```
lab9 git:(master) ✗ diff main.cpp ../lab7/main.cpp
4d3
< #include <omp.h>
99d97
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
104d101
<         #pragma omp parallel for private(i, j, k) shared(data)
120d116
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
125d120
<         #pragma omp parallel for private(i, j, k) shared(data)
141d135
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
146d139
<         #pragma omp parallel for private(i, j, k) shared(data)
162d154
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
167d158
<         #pragma omp parallel for private(i, j, k) shared(data)
185d175
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
190d179
<         #pragma omp parallel for private(i, j, k) shared(data)
206d194
<         #pragma omp parallel for private(i, j, k) shared(data, buff)
211d198
```

```
<           #pragma omp parallel for private(i, j, k) shared(data)
220c207
<       #pragma omp parallel for private(i, j, k) shared(data, next) reduction(max: diff)
```

Результаты

В данной лабораторной работе мы можем сравнить с результатом программы из 7 Лр.

1.

	MPI	MPI + OpenMP
2 2 2, 20 20 20	5006.32ms	4134.12ms
2 2 4, 20 20 10	5894.24ms	4323.8ms

По результатам мы можем понять, что увеличения по времени есть, но лучше проводить сравнения на кластере машин и с большим кол-вом входных данных.

Выводы

Данная ЛР была лучшей из всех, ведь написав 7 ЛР, я очень обрадовался, когда узнал, что не зря мучился с ней, так как 9 ЛР имеет немного отличий. Превосходный интерфейс OpenMP позволил сдать мне данную работу быстрее всего, тем более мне ещё и попался легкий вариант. Единственное, что заставило меня немного подумать, это таймлимит. Интересная особенность, что лучше не паралеллить все что есть в программе, так как поддержка параллелизма не бесценна для процессора.