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: Валов В.В
Группа: М8О-408Б
Преподаватели: А. Ю. Морозов,
К. Г. Крашенинников

Москва, 2022

Условие

Цель работы:

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

Вариант *Распараллеливание основных циклов через parallel for*

Программное и аппаратное обеспечение

GPU:
Compute capability: 7.5;
Графическая память: 4294967296;
Разделяемая память: 49152;
Константная память: 65536;
Количество регистров на блок: 65536;
Максимальное количество блоков: (2147483647, 65535, 65535);
Максимальное количество нитей: (1024, 1024, 64);
Количество мультипроцессоров: 6.
Сведения о системе:
Процессор: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
Память: 16,0 ГБ;
HDD: 237 ГБ.
Программное обеспечение:
OS: Windows 10;
IDE: Visual Studio 2019;
Компилятор: nvcc.

Метод решения

Для решения задачи сетка делилась на области. За каждую область отвечал один процесс. Сначала происходит обмен граничными данными между процессами, потом обновляются значения во всех ячейках. Потом происходит вычисление погрешности, по результатам которого вычисления останавливаются или продолжаются. В данной работе, в отличие от седьмой обмен данными между процессами происходит с

помощью созданных мной типов sendrec_lr и sendrec_up типа hvector. Для вывода данных используется составной тип out_type также типа hvector.

Описание программы

Передача данных всем процессам:

```
MPI_Bcast(&bc_left, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&bc_right, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&bc_up, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&bc_down, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&bc_front, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&bc_back, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&lx, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&ly, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&lz, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&nX, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&nY, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&nZ, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&nb_X, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&nb_Y, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&nb_Z, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&epsilon, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast(&init_v, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Инициализация переменных для составных типов:

```
int sendrec_left_right_part = nY + 2;
int sendrec_up_down_part = nX + 2;
int out_type_minot_part1 = nY;
int out_type_minot_part2 = n_size * nX;
int out_type_minot_part3 = n_size * nX * nb_X;
unsigned int size = n_size * nX * nY;
 MPI_Aint format = n_size * nX * nb_X * nY * jb + n_size * nX * ib;
```

Инициализация типов:

```
MPI_Datatype sendrec_left_right;
MPI_Datatype sendrec_up_down;
MPI_Datatype out_type;
MPI_File output_file;
```

Создание типов:

```
MPI_Type_vector(sendrec_left_right_part, 1, sendrec_up_down_part, MPI_DOUBLE,
&sendrec_left_right);
        MPI_Type_vector(1, sendrec_up_down_part, 0, MPI_DOUBLE,
&sendrec_up_down);
MPI_Type_create_hvector(out_type_minot_part1, out_type_minot_part2,
out_type_minot_part3, MPI_CHAR, &out_type);
```

Запись типов:

```
        MPI_Type_commit(&sendrec_left_right);
         MPI_Type_commit(&sendrec_up_down);
```

Запись в файл:

```
MPI_Type_commit(&out_type);
         MPI_File_delete(output_file_name, MPI_INFO_NULL);
        MPI_File_open(MPI_COMM_WORLD, output_file_name, MPI_MODE_CREATE |
MPI_MODE_WRONLY, MPI_INFO_NULL, &output_file);
        MPI_File_set_view(output_file, format, MPI_CHAR, out_type, "native",
MPI_INFO_NULL);
MPI_File_write_all(output_file, text, size, MPI_CHAR, &status);
MPI_File_close(&output_file);
```

Результаты:

MPI+OMP	1	2
8	0.0009	0.0021
16	0.0010	0.0093
32	0.0801	0.0692
64	0.1004	0.4391

Выводы

Технология MPI позволяет распараллеливать вычисления, за счет этого возрастает скорость работы программы. Технология OpenMP позволяет выполнять все циклы параллельно, это увеличивает скорость работы программы еще сильнее. Но есть и минусы, написание такого кода занимает много времени и сделать это не очень просто. Я использовал операцию Bsend для отправки данных, она является неблокирующей за счет буфера, это позволяет упростить код.