Домашнее Задание №3
Липатов Данила Вячеславович
МСМТ 243

Ссылка на GIT

I пункт.

Для запуска на кластере был скомпилирован код следующей командой

```
mpicc hw_1_mpi.c -o hw_1_mpi.out -lm
```

Предварительно подгрузив все необходимые модули:

```
module load INTEL/oneAPI_2022
module load nvidia_sdk/nvhpc/23.5
```

В целом, скрипт выполнился хорошо с маленькой погрешность. На рис. 1 предоставлено изображение для 11 точек:

x	Exact Solution	Approximate Solution	Absolute Error
0.000000	0.000000	0.000000	0.000000
0.100000	0.146595	0.146690	0.000094
0.200000	0.278875	0.279539	0.000664
0.300000	0.383898	0.383936	0.000039
0.400000	0.451355	0.452183	0.000828
0.500000	0.474605	0.474494	0.000111
0.600000	0.451355	0.452183	0.000828
0.700000	0.383898	0.383936	0.000039
0.800000	0.278875	0.279539	0.000664
0.900000	0.146595	0.146690	0.000094
1.000000	0.000000	0.000000	0.000000

рис. 1

II пункт.

Аналогично предыдущему пункту через команду

```
srun -n 1 --constrain="type_a" hw_1_mpi.out
```

Запускались для разных N и разных n от 1 до 24 выполнение скриптов

Графики зависимости n от времени предоставлен на рис. 2 и на рис.3 gflops от n.

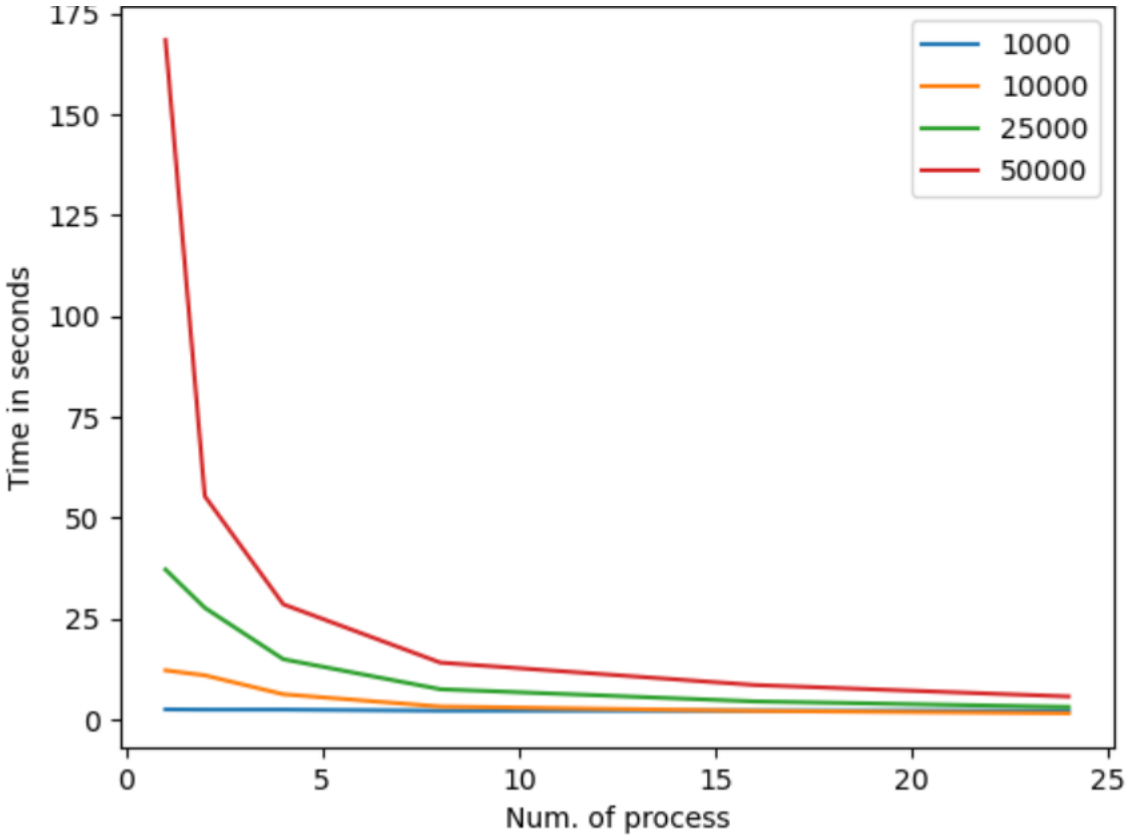

рис. 2

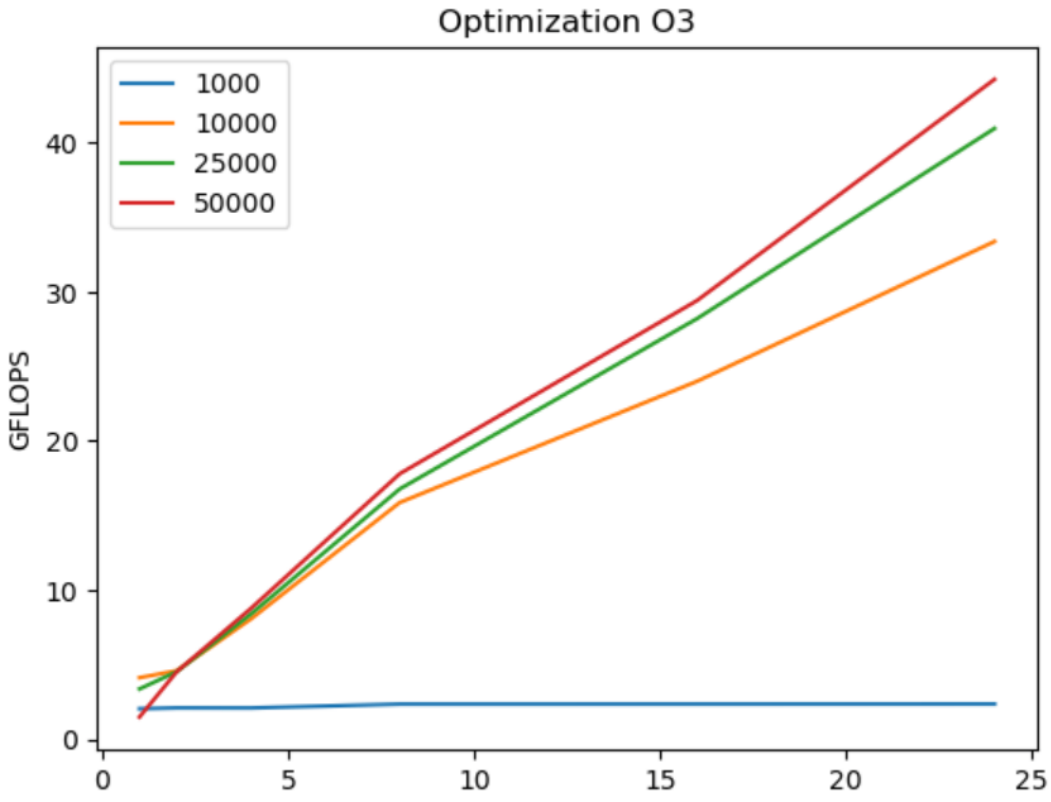

рис.3

III пункт.

Для асинхронного выполнения команды MPI_Sendrecv были заменены на MPI_Isend, MPI_Irecv и так же добавлен барьер

```
MPI_Waitall(request_count, requests, MPI_STATUSES_IGNORE);
```

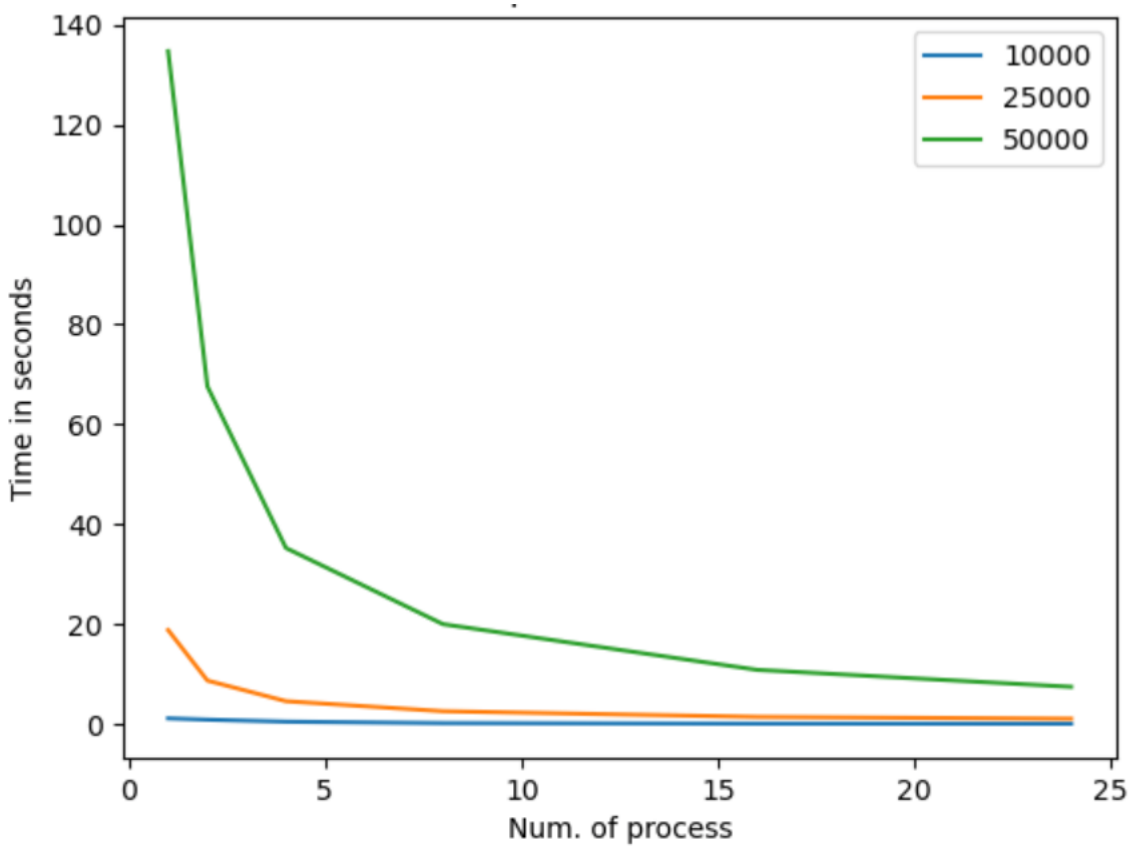

рис. 4

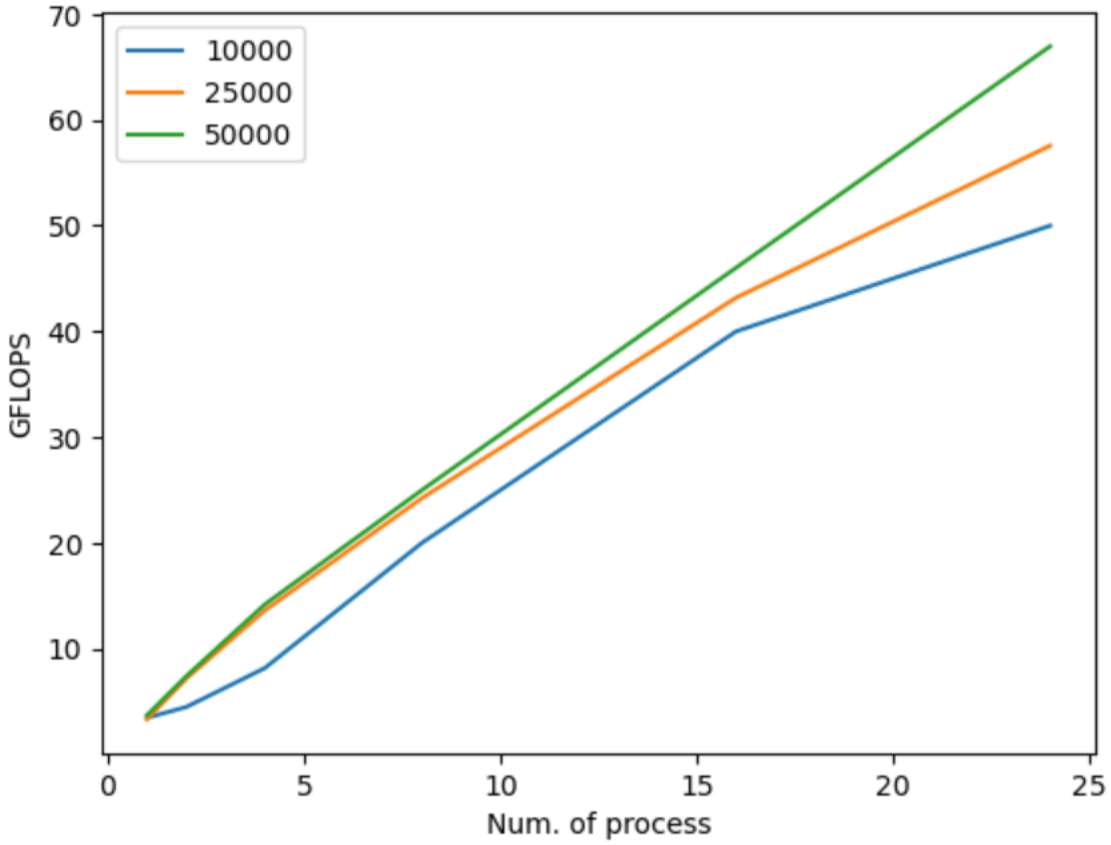

рис. 5

IV Пункт.

Помимо MPI_Gatherv() был так же добавлен MPI_Scatter()
Создаёт массив начальных температур. Этот массив рассылается другим процессам с помощью MPI_Scatter.

В целом, сходимость с предыдущими реализациями такая же

```
Average elapsed time over 10 runs: 17.20419 seconds
Average RMSE over 10 runs: 0.0000002841
```

Пример для 25000 точек.

V пункт

Наиболее трудным моментом является компиляция с CUDA:

nvcc -o hw_1_mpi_cuda hw_1_mpi_cuda.cu -ccbin mpicc -lcudart -lm

Где **`-ccbin mpicc`**

Указывает, какой компилятор должен использоваться для обработки CPU-кода.

-lcudart

Опция для линковки с библиотекой CUDA. Эта библиотека предоставляет функции, необходимые для работы программы с GPU.

srun -n 1 --gres=gpu:1 --constraint="type_a" hw_1_mpi_cuda.out

```
ремя выполнения: 0.81431 секунд
очное решение:
=0.00, u=0.00000     0.00000
=0.01, u=0.01491     0.00001
=0.02, u=0.02981     0.00001
=0.03, u=0.04468     0.00004
=0.04, u=0.05950     0.00003
=0.05, u=0.07426     0.00007
=0.06, u=0.08895     0.00004
=0.07, u=0.10356     0.00010
=0.08, u=0.11806     0.00006
=0.09, u=0.13244     0.00013
=0.10, u=0.14669     0.00007
=0.11, u=0.16080     0.00016
=0.12, u=0.17475     0.00009
=0.13, u=0.18852     0.00019
=0.14, u=0.20211     0.00010
=0.15, u=0.21550     0.00021
=0.16, u=0.22867     0.00011
=0.17, u=0.24162     0.00024
=0.18, u=0.25433     0.00013
=0.19, u=0.26679     0.00026
=0.20, u=0.27899     0.00014
=0.21, u=0.29091     0.00029
=0.22, u=0.30254     0.00015
=0.23, u=0.31387     0.00031
=0.24, u=0.32489     0.00016
=0.25, u=0.33560     0.00033
=0.26, u=0.34597     0.00017
=0.27, u=0.35600     0.00035
=0.28, u=0.36567     0.00018
```

Рис. 7 (пример)