

Теория параллелизма

Отчёт

Уравнение теплопроводности на CUDA

Выполнил Грищенко Александр Михайлович, 21932

05.2023

1 Цели работы

Реализовать решение уравнение теплопроводности (пятиточечный шаблон) в двумерной области на равномерных сетках.

Перенести программу на GPU используя CUDA. Операцию редукции (подсчет максимальной ошибки) реализовать с использованием библиотеки CUB.

Сравнить скорость работы для разных размеров сеток на графическом процессоре с предыдущими реализациями на OpenACC и OpenACC с cuBLAS.

Произвести профилирование программы и оптимизацию кода.

2 Используемый ускоритель

NVIDIA GeForce RTX 2080 Ti

3 Используемый компилятор

nvcc: NVIDIA (R) Cuda compiler driver V11.0.221

Для компиляции:

```
/usr/local/cuda/bin/nvcc task_4.cu -o task4
```

Для запуска:

```
./task4 -s 128 -i 1e6 -a 1e-6
```

Порядок параметров не важен, как и сами параметры. По умолчанию размер сетки — 128, количество итераций — 10^6 , точность — 10^{-6}

4 Используемый профилировщик

nsys: NVIDIA Nsight Systems version 2022.4.1.21-0db2c85 с флагом `--trace cuda`

```
nsys profile -t cuda ./task4 -s 512 -i 1000
```

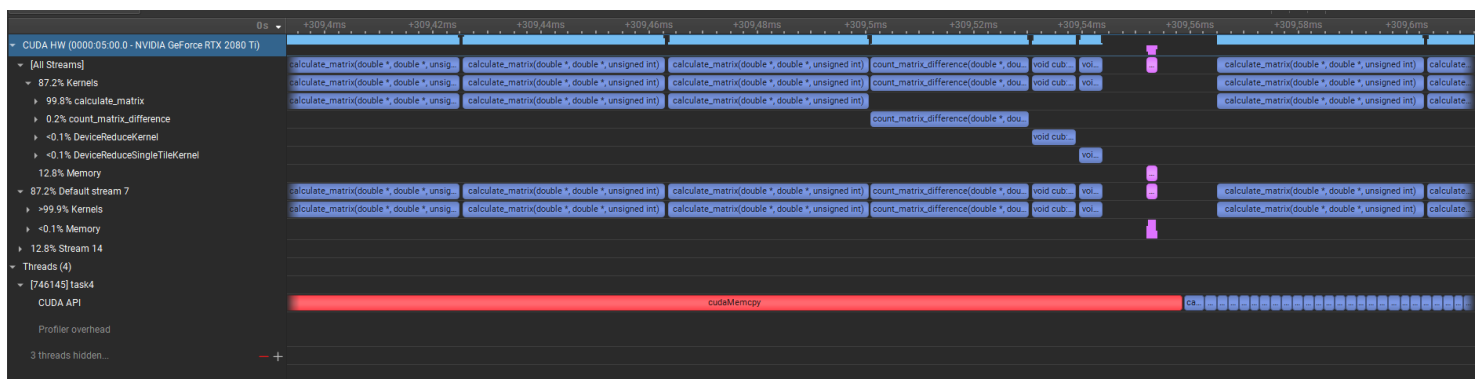
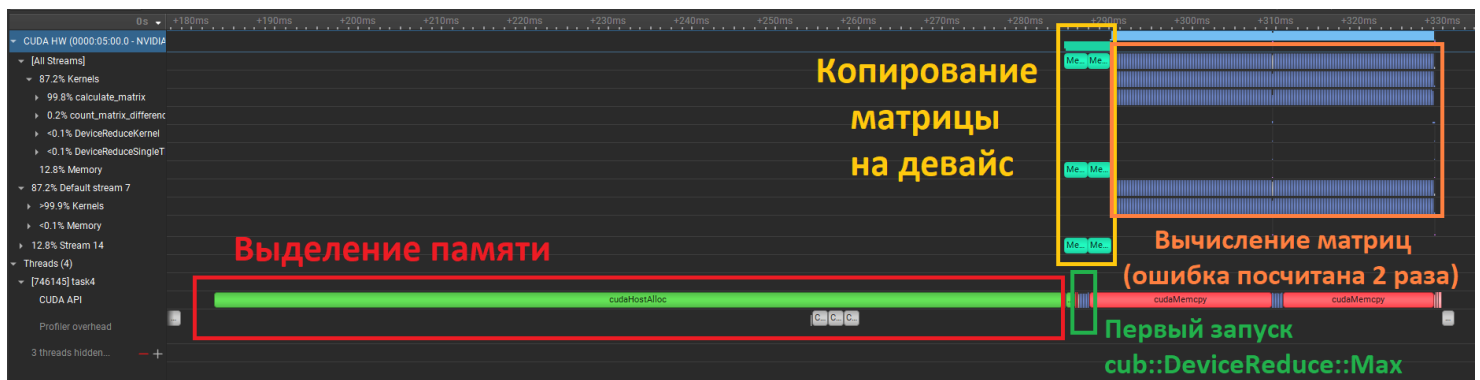
5 Замер времени работы

Для замера времени работы использовалась библиотека *chrono*. Замер времени производился несколько раз, затем бралось среднее время.

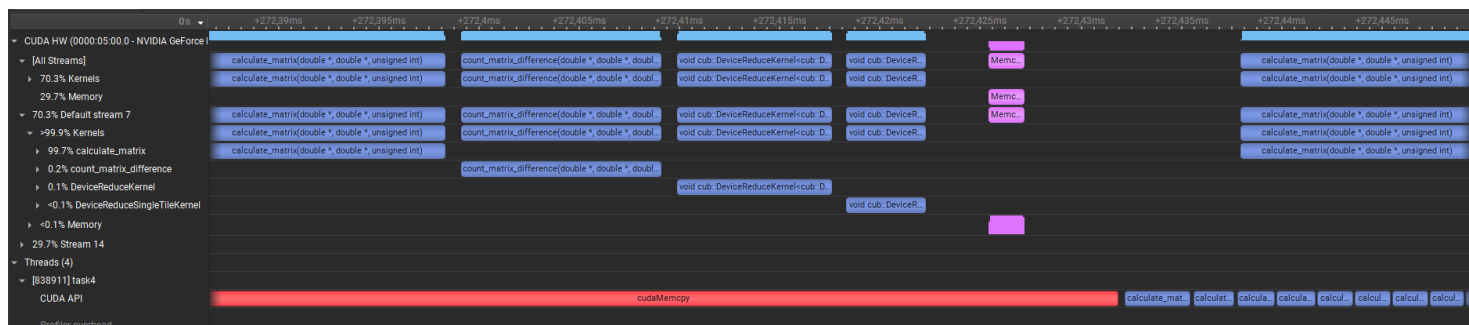
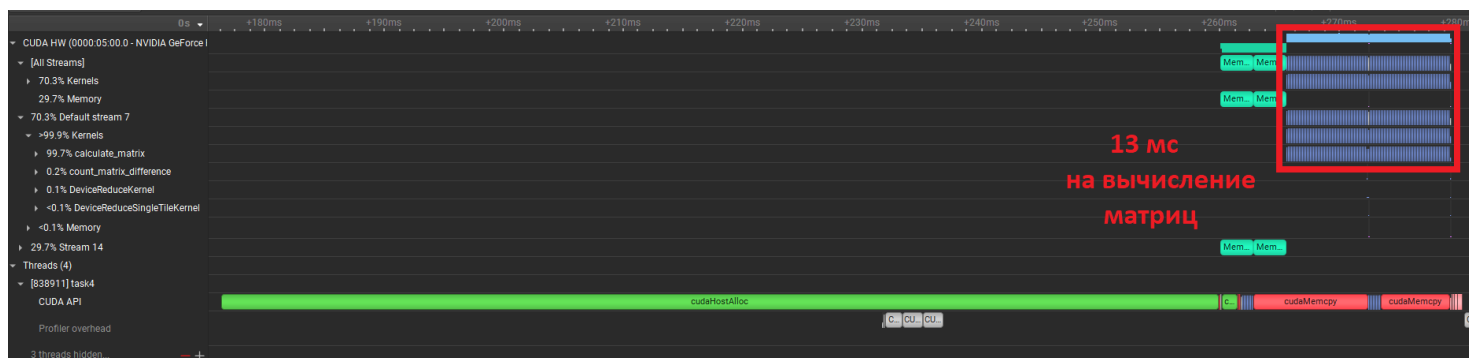
6 Этапы оптимизации на сетке 512*512

Этап №	Время выполнения, с	Точность	Количество операций	Комментарии (что было сделано)
1	0.1	0.01	1000	Код переписан на CUDA, параметры ядра: <code><<<net_size, net_size>>></code>
2	0.075	0.01	1000	Оптимальные параметры ядра
3	0.075	0.01	1000	Переход на cuda graph, swar заменен на 2 подсчета

6.1 Этап 1

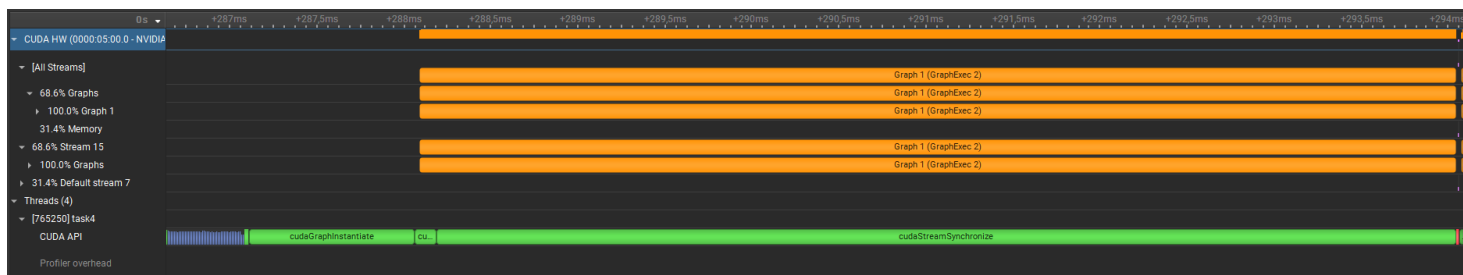
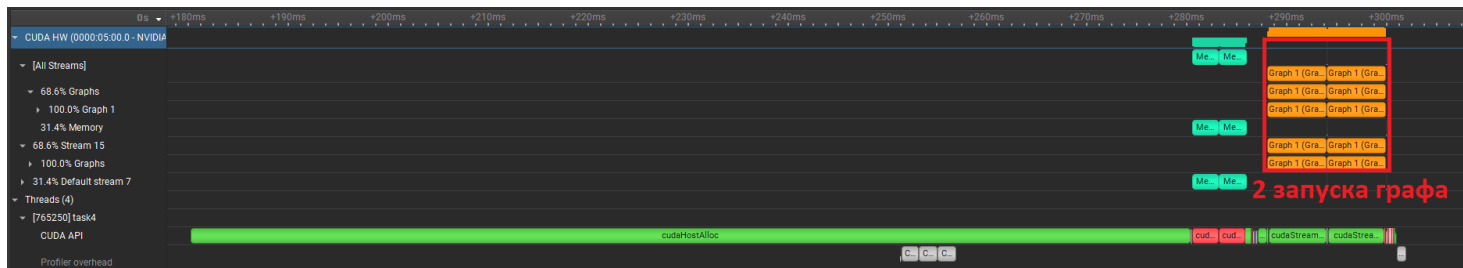


6.2 Этап 2



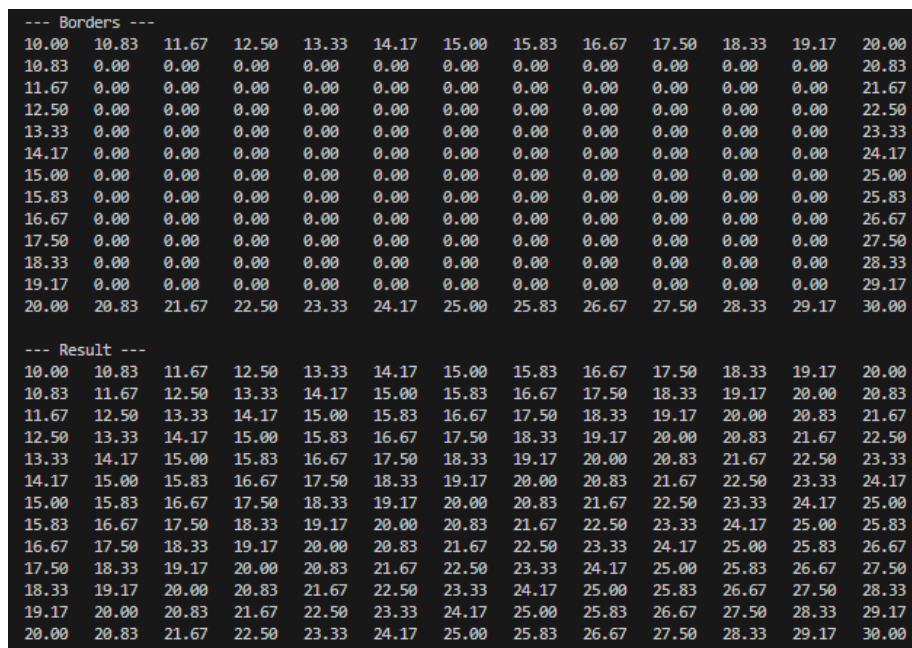
На вычисление матриц суммарно ушло 13 мс, что более чем в 2 раза меньше, чем на прошлом этапе.

6.3 Этап 3



Переход на граф никак не ускорил работу, но теперь итерации отображаются как отдельные блоки.

7 Проверка правильности работы программы

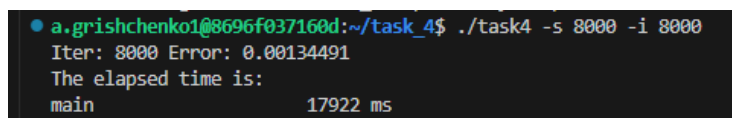


The image shows a terminal window displaying a 13x13 matrix. The matrix is divided into two sections: '--- Borders ---' and '--- Result ---'. The 'Borders' section shows a matrix where the outer boundary is filled with values, and the inner cells are zero. The 'Result' section shows the matrix after a program has processed it, where the values are shifted and the boundary is now zero.

--- Borders ---												
10.00	10.83	11.67	12.50	13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00
10.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20.83
11.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	21.67
12.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	22.50
13.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.33
14.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	24.17
15.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	25.00
15.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	25.83
16.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	26.67
17.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	27.50
18.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	28.33
19.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	29.17
20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67	27.50	28.33	29.17	30.00

--- Result ---												
10.00	10.83	11.67	12.50	13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00
10.83	11.67	12.50	13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83
11.67	12.50	13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67
12.50	13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50
13.33	14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33
14.17	15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17
15.00	15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00
15.83	16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83
16.67	17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67
17.50	18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67	27.50
18.33	19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67	27.50	28.33
19.17	20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67	27.50	28.33	29.17
20.00	20.83	21.67	22.50	23.33	24.17	25.00	25.83	26.67	27.50	28.33	29.17	30.00

Рис. 1: Скриншот массива 13*13 после заполнения границ и после работы всей программы



The image shows a terminal window with the following text:

```
● a.grishchenko1@8696f037160d:~/task_4$ ./task4 -s 8000 -i 8000
Iter: 8000 Error: 0.00134491
The elapsed time is:
main 17922 ms
```

Рис. 2: Сетка 8000*8000

8 Сравнение времени работы с предыдущими реализациями

Количество операций во всех реализациях кратно размеру сетки.

8.1 CPU-onecore

Размер сетки	Время выполнения, с	Точность	Количество операций
128*128	0.15	9.9e-07	30080
256*256	2.2	9.9e-07	102912
512*512	28	9.9e-07	339968

8.2 CPU-multicore

Размер сетки	Время выполнения, с	Точность	Количество операций
128*128	0.2	9.9e-07	30080
256*256	2.2	1e-07	102912
512*512	25	9.9e-07	339968
1024*1024	334	9.89e-07	1000448

8.3 GPU (OpenACC)

Размер сетки	Время выполнения, с	Точность	Количество операций
128*128	0.18	9.8e-07	30080
256*256	0.37	9.9e-07	102912
512*512	2.5	9.9e-07	339968
1024*1024	34	1.4e-06	1000448

8.4 GPU (OpenACC + cuBLAS)

Размер сетки	Время выполнения, с	Точность	Количество операций
128*128	0.6	9.8e-07	30080
256*256	0.95	9.9e-07	102912
512*512	3	9.9e-07	339968
1024*1024	34	1.4e-06	1000448

8.5 GPU (CUDA)

Размер сетки	Время выполнения, с	Точность	Количество операций
128*128	0.14	9.6e-07	30080
256*256	0.34	9.8e-07	102912
512*512	2.9	9.8e-07	339968
1024*1024	35.5	1.4e-06	1000448

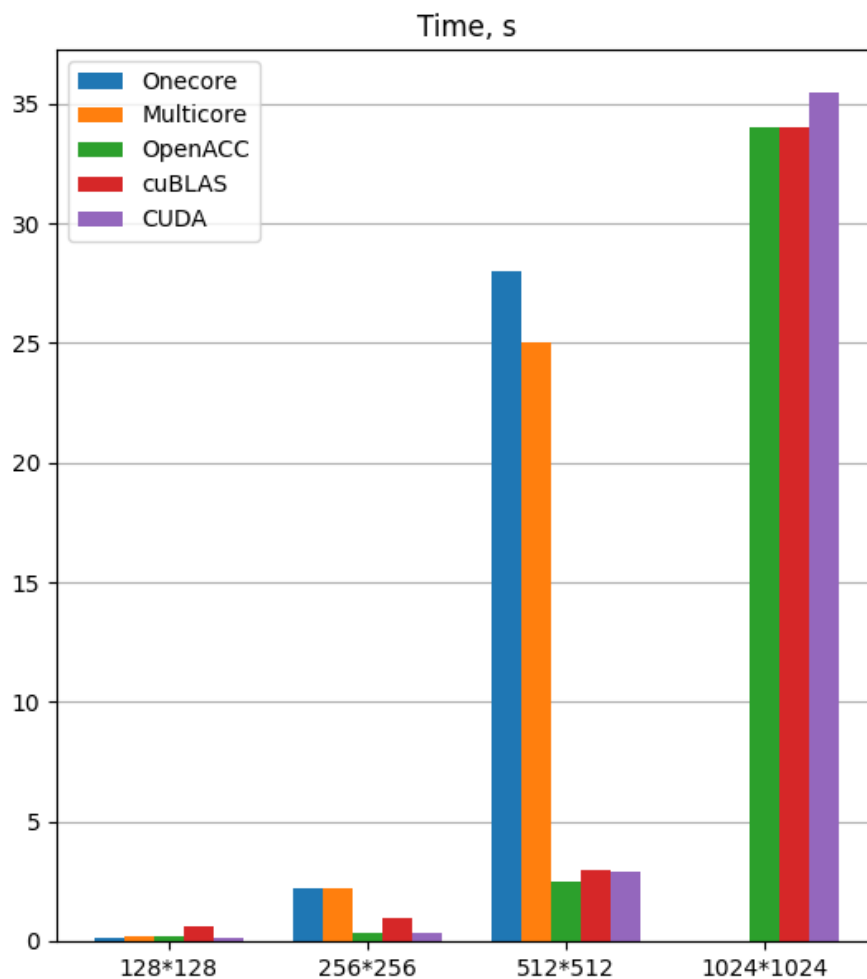


Рис. 3: Диаграмма сравнения времени работы

9 Сравнение реализаций CUDA и OpenACC + cuBLAS

Для сравнения запустим профилирование на размере сетки 512 с точностью $1e-6$ и количеством итераций $1e6$.

Как можно заметить, cuBLAS делает работу по подсчету быстрее, чем CUDA, но слишком долгое создание контекста не даёт опередить CUDA в целом.

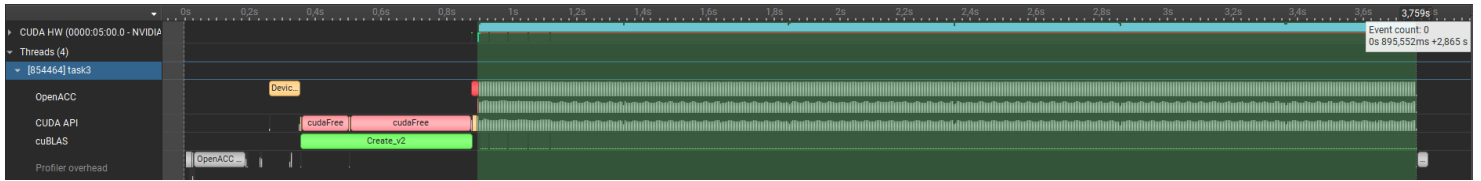


Рис. 4: cuBLAS в Nsight Systems

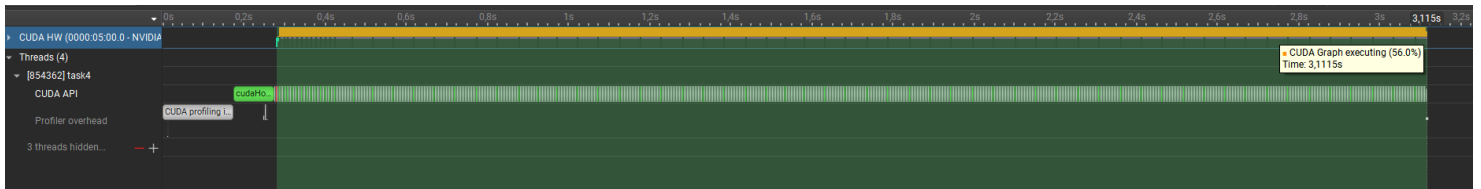


Рис. 5: CUDA в Nsight Systems

10 Вывод

CUDA показывает себя хорошо на маленьких стеках, на больших же она почти не уступает другим реализациями, что делает реализацию на CUDA более универсальной.

11 Приложение

[Ссылка на GitHub](#)