### Теория Параллелизма

#### Отчет

# Решения уравнения теплопроводности методом разностной схемы (CUDA)

Выполнил Абрамов Никита, группа 23931

Цель: Реализовать решение уравнения теплопроводности в двумерной области с использованием разностной схемы на равномерных сетках. Программа должна учитывать линейную интерполяцию на границах и значения в углах. Учесть точность 10^-6 и максимальное число итераций 10^6. Реализовать программу на С++ с использованием CUDA. Сравнить производительность с прошлыми реализациями, провести профилирование и сделать вывод об использовании CUDA.

Используемый компилятор: pgc++ (aka nvc++) 23.11-0 64-bit

Используемый профилировщик: NVIDIA Nsight systems

Замеры производились с помощью библиотеки <chrono>

## Выполнение на GPU (OpenACC)

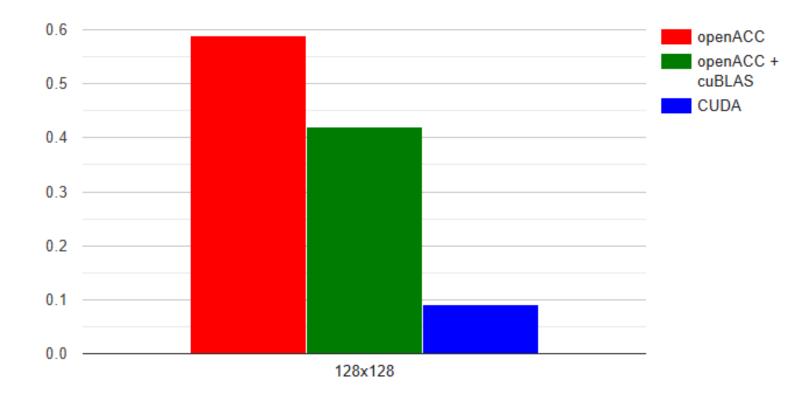
Размер сетки	Время выполнения (c)	Точность	Количество итераций
128x128	0,59	1e-6	40 000
256x256	1,37	1e-6	110 000
512x512	512x512 4,44		340 000
1024x1024	33,95	1e-6	1 000 000

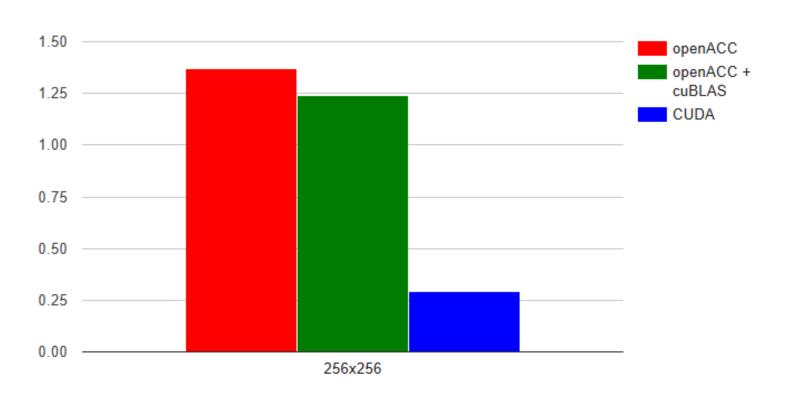
## Выполнение на GPU (OpenACC + cuBLAS)

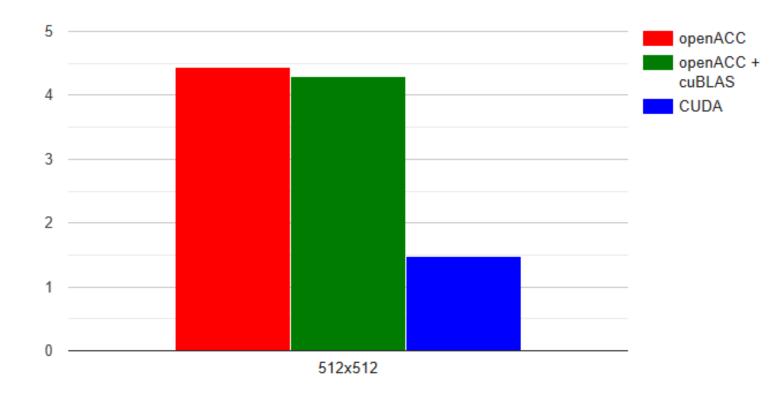
Размер сетки	Время выполнения (c)	Точность	Количество итераций
128x128	0,42	1e-6	40 000
256x256	1,24	1e-6	110 000
512x512	4,30	1e-6	340 000
1024x1024	33,42	1e-6	1 000 000

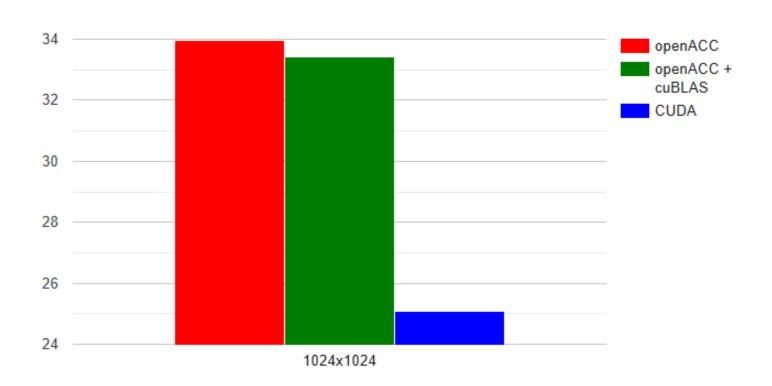
## Выполнение на GPU (CUDA)

Размер сетки	Время выполнения (c)	Точность	Количество итераций
128x128	0,09	1e-6	40 000
256x256	0,29	1e-6	110 000
512x512	1,48	1e-6	340 000
1024x1024	25,10	1e-6	1 000 000









jacobi_profile.nsys-rep ×								
□ Timeline View	® Options	2					■ Q □ 1x	▲ 2 warnings, 10 messages
	1s •	s +45,975ms +45,98ms +45,985ms	+45,99ms +45,995ms 1s 46,0	0007ms +46,005ms +46,01ms	+46,015ms +46,02ms	+46,025ms +46,03ms +46,035ms	+46,04ms +46,045ms +46,0	05ms +46,055ms -
▼ CUDA HW (0000:89:00.0 -	Gra Tesla V Kerr Memo	iel ny						
▶ [All Streams]		updateKernel(double *, double *, int)	updateKerne	el(double *, double *, int)		updateKernel(double *, double *, int)	updateKernel(dout	ole *, double *, int)
▼ >99,9% Default stream 7		updateKernel(double *, double *, int)	updateKerne	el(double *, double *, int)		updateKernel(double *, double *, int)	updateKernel(doub	ole *, double *, int)
>99,9% Kernels		updateKernel(double *, double *, int)	updateKerne	el(double *, double *, int)		updateKernel(double *, double *, int)	updateKernel(dout	ole *, double *, int)
> <0,1% Memory								
> <0,1% Stream 16								
→ Threads (6)								
▼ [856830] a.out	0 to 10	%						
OS runtime libraries								
CUDA API		updateKernel		updateKernel		updateKernel		updateKernel
Profiler overhead								
5 threads hidden	— + 0 to 10	96						
		4		:				) ·

#### Вывод:

Использование CUDA дало ускорение более чем в два-три раза на сетках 128, 256, 512 и увеличило время с 33 секунд до 25 секунд на сетке 1024х1024

За счет точного контроля над ядрами CUDA (свои ядра + настройка grid, block) удалось добиться уменьшения времени выполнения ядер и уменьшения интервала между самими вызовами этих ядер, что способствовало общему уменьшению времени выполнения программы.