Теория параллелизма

Отчет

Уравнение теплопроводности (Разностная схема)

Выполнил Михайлюк Алексей Александрович, 22930

30.05.2024

Цель работы: реализовать решение уравнения теплопроводности в двумерной области с использованием разностной схемы и пятиточечного шаблона на равномерных сетках размером (128^2, 256^2, 512^2, 1024^2). Использовать компилятор pgcc/pgc++ с ключами "-acc" и "-Minfo=all" для сборки программы и перенести ее на GPU с помощью директив OpenACC. При замерах времени на CPU собрать данные об использовании нескольких ядер (acc=multicore). Произвести профилирование программы с использованием "Nsight Systems" и провести оптимизацию кода.
Используемый компилятор: pgc++.

Используемый профилировщик: “Nsight Systems”.

Как производили замер времени работы: для замера времени работы программы использовал библиотеку.

auto start = std::chrono::high_resolution_clock::now(); {

Участок кода, где выполняется вычисление новых значений внутренних точек матрицы curmatrix на основе значений соседних точек из матрицы prevmatrix.

} auto end = std::chrono::high_resolution_clock::now();

auto time_s = std::chrono::duration_cast(end - start).count();

Выполнение на CPU

CPU-onecore

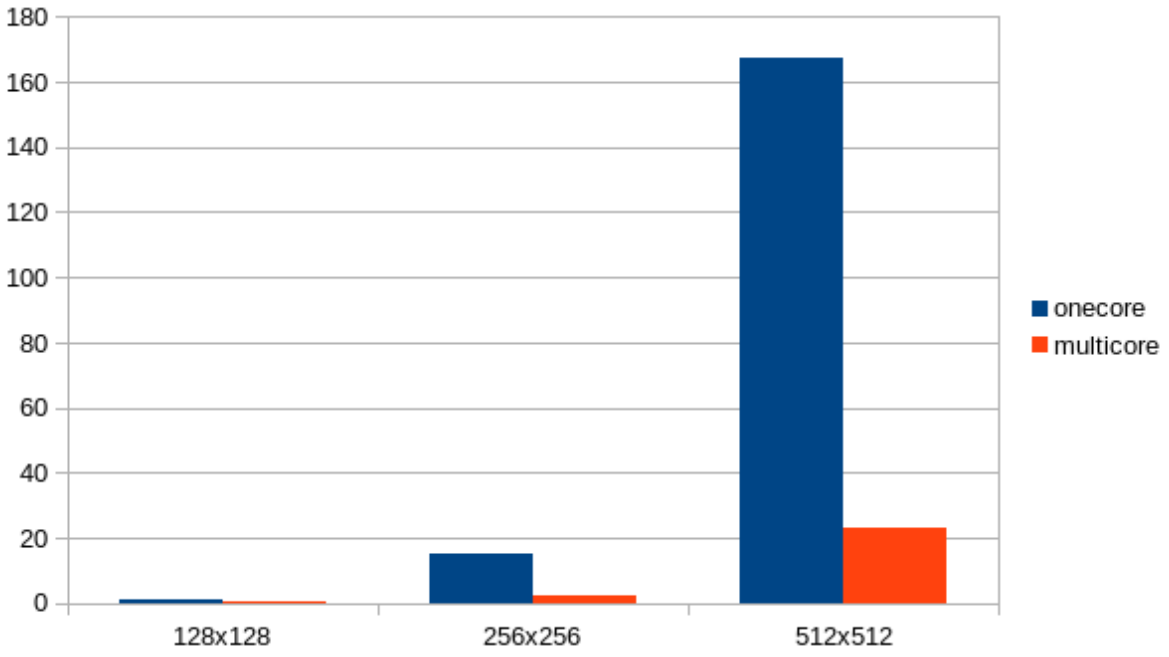
Размер сетки	Время выполнения, в	Точность	Количество итераций

	секундах		
128x128	0.923	1e-6	37000
256x256	15.402	1e-6	125000
512x512	167.607	1e-6	410000
1024x1024	1913.046	1e-6	1275000

CPU-multicore

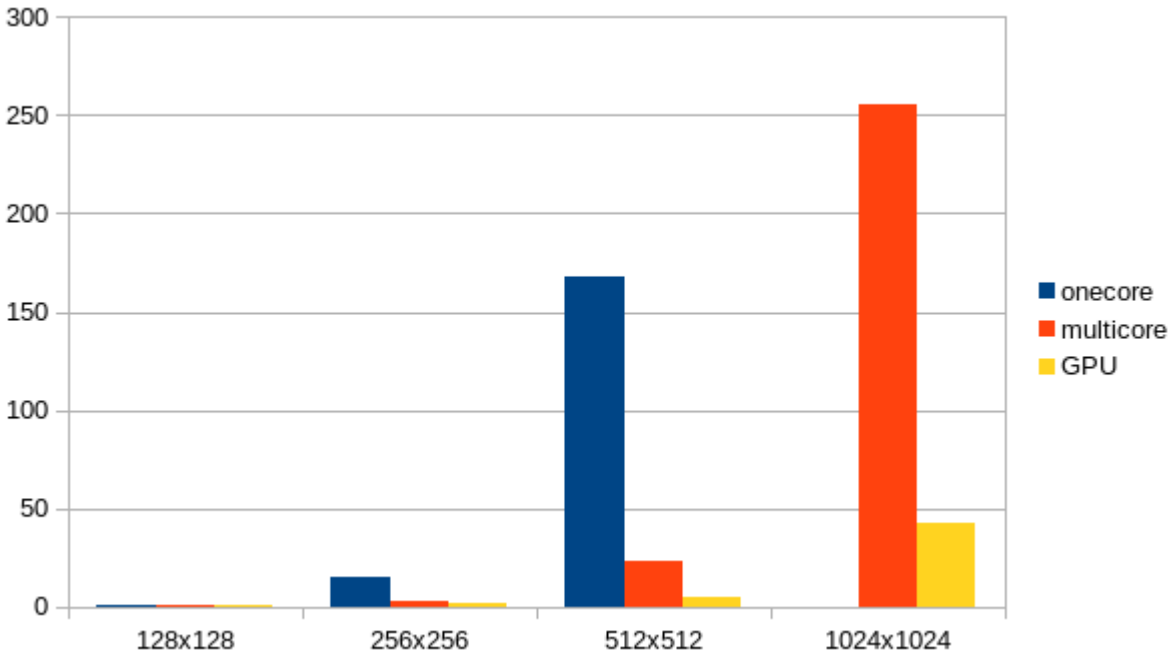
Размер сетки	Время выполнения, в секундах	Точность	Количество итераций
128x128	0.438	1e-6	37000
256x256	2.424	1e-6	125000
512x512	22.993	1e-6	410000
1024x1024	255.227	1e-6	1275000

Диаграмма сравнения время работы CPU-one и CPU-multi

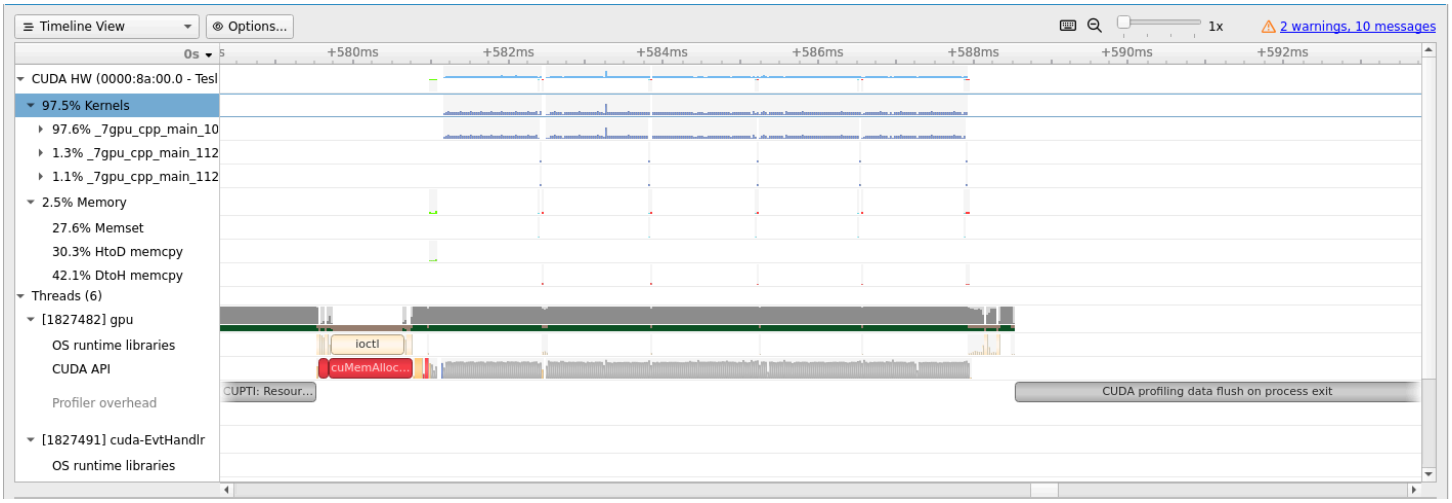


Выполнение на GPU

Размер сетки	Время выполнения, в секундах	Точность	Количество итераций
128x128	0.384	1e-6	36700
256x256	1.400	1e-6	125000
512x512	5.367	1e-6	409800
1024x1024	42.845	1e-6	1274200



Оптимизации: Использовальзовал синхронизацию данных между CPU и GPU, Благодаря чему получилось добиться значений: 97,5% времени затрачено на вычисления и 2,5% на синхронизацию данных между CPU и GPU. Также при использовании директивы openACC, я добавил такие атрибуты как independed и collapse(2), первый атрибут указывает, что итерации цикла независимы друг от друга, это позволяет компилятору эффективнее распределить итерации этого цикла по доступным потокам, второй атрибут схлопывает два цикла в один, создавая более крупное пространство итераций одним циклом, которое можно эффективнее распараллелить.



Вывод:

На больших матрицах GPU показывает наилучший результат.