МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №1
по курсу «Программирование графических процессоров»

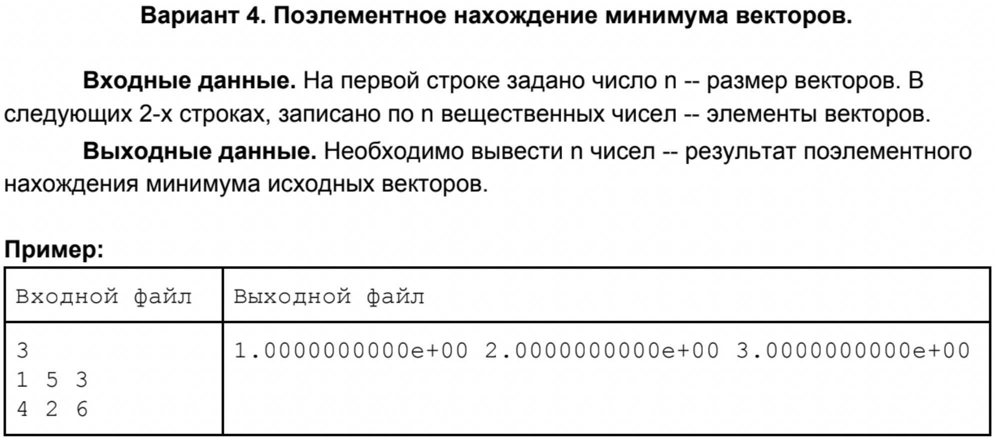
Освоение программного обеспечения для работы с технологией CUDA. Примитивные операции над векторами.

Выполнил: Полей-Добронравова
Амелия
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы. Ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений(CUDA). Реализация одной из примитивных операций над векторами.



Вариант 4. Поэлементное нахождение минимума векторов.

Входные данные. На первой строке задано число n -- размер векторов. В следующих 2-х строках, записано по n вещественных чисел -- элементы векторов.

Выходные данные. Необходимо вывести n чисел -- результат поэлементного нахождения минимума исходных векторов.

Пример:

Входной файл	Выходной файл

3
1 5 3
4 2 6
	1.0000000000e+00 2.0000000000e+00 3.0000000000e+00

Программное и аппаратное обеспечение

Компилятор nvcc версии 7.0(g++ версии 4.8.4) на 64-х битной Ubuntu 14.04 LTS. Параметры графического процессора:

Compute capability : 6.1
Name : GeForce GTX 1050
Total Global Memory : 2096103424
Shared memory per block : 49152
Registers per block : 65536
Max threads per block : (1024, 1024, 64)
Max block : (2147483647, 65535, 65535)
Total constant memory : 65536
Multiprocessors count : 5

Метод решения

Ввод данных осуществляется в main, который работает на CPU. Далее заводятся три массива, данные которых будут храниться на стороне GPU, потому что GPU не сможет работать с памятью CPU. Первые два массива - копии введенных, в третий массив будет сохраняться результат обработки. В функции kernel на GPU идет распараллеливание обработки массива. Из-за того, что потоков может не хватить на весь размер n, каждый поток обрабатывает не по одному i-тому элементу, а по несколько через offset - шаг, общее число потоков. Индекс i с которого начинается обработка - абсолютный номер потока.

Далее по завершении обработки данные из результата dev_v3 записываются в v1, и выводятся отформатировано на экран через пробел.

Описание программы

Макрос **CSC** - макрос для отслеживания ошибок со стороны GPU, вызывается около функций для cuda и выводит текст ошибки при cudaError_t не равным cudaSuccess.
kernel - обработка массивов по потокам gpu.
main - ввод данных, подготовка для передачи данных kernel для обработки, вывод результата.

Результаты

Работа с GPU на разных конфигурациях ядер:

Тест:	Результаты в милисекундах:

3
1 2 3
4 5 6
	kernel = <<<1, 32>>>, time = 0.025568
	kernel = <<<1, 64>>>, time = 0.010528
	kernel = <<<1, 128>>>, time = 0.008864
	kernel = <<<1, 256>>>, time = 0.008576
	kernel = <<<1, 512>>>, time = 0.008960
	kernel = <<<1, 1024>>>, time = 0.008768
	kernel = <<<2, 32>>>, time = 0.008640
	kernel = <<<2, 64>>>, time = 0.008896
	kernel = <<<2, 128>>>, time = 0.008288
	kernel = <<<2, 256>>>, time = 0.008672
	kernel = <<<2, 512>>>, time = 0.008864
	kernel = <<<2, 1024>>>, time = 0.008448
	kernel = <<<4, 32>>>, time = 0.008416
	kernel = <<<4, 64>>>, time = 0.008608
	kernel = <<<4, 128>>>, time = 0.008448
	kernel = <<<4, 256>>>, time = 0.008480
	kernel = <<<4, 512>>>, time = 0.008960
	kernel = <<<4, 1024>>>, time = 0.008704
	kernel = <<<8, 32>>>, time = 0.008480
	kernel = <<<8, 64>>>, time = 0.008384
	kernel = <<<8, 128>>>, time = 0.008448
	kernel = <<<8, 256>>>, time = 0.008448
	kernel = <<<8, 512>>>, time = 0.008928
	kernel = <<<8, 1024>>>, time = 0.008960
	kernel = <<<16, 32>>>, time = 0.008480
	kernel = <<<16, 64>>>, time = 0.008544
	kernel = <<<16, 128>>>, time = 0.008448
	kernel = <<<16, 256>>>, time = 0.008512
	kernel = <<<16, 512>>>, time = 0.009024
	kernel = <<<16, 1024>>>, time = 0.010336
	kernel = <<<32, 32>>>, time = 0.008416
	kernel = <<<32, 64>>>, time = 0.008480
	kernel = <<<32, 128>>>, time = 0.008448
	kernel = <<<32, 256>>>, time = 0.008736
	kernel = <<<32, 512>>>, time = 0.009888
	kernel = <<<32, 1024>>>, time = 0.012512
	kernel = <<<64, 32>>>, time = 0.008768
	kernel = <<<64, 64>>>, time = 0.008768
	kernel = <<<64, 128>>>, time = 0.008928
	kernel = <<<64, 256>>>, time = 0.009440
	kernel = <<<64, 512>>>, time = 0.011200
	kernel = <<<64, 1024>>>, time = 0.017664
	kernel = <<<128, 32>>>, time = 0.009536

	kernel = <<<128, 64>>>, time = 0.009472
	kernel = <<<128, 128>>>, time = 0.009664
	kernel = <<<128, 256>>>, time = 0.010304
	kernel = <<<128, 512>>>, time = 0.013696
	kernel = <<<128, 1024>>>, time = 0.026656
	kernel = <<<256, 32>>>, time = 0.011264
	kernel = <<<256, 64>>>, time = 0.011104
	kernel = <<<256, 128>>>, time = 0.011264
	kernel = <<<256, 256>>>, time = 0.013088
	kernel = <<<256, 512>>>, time = 0.018976
	kernel = <<<256, 1024>>>, time = 0.045920
	kernel = <<<512, 32>>>, time = 0.014176
	kernel = <<<512, 64>>>, time = 0.014368
	kernel = <<<512, 128>>>, time = 0.014656
	kernel = <<<512, 256>>>, time = 0.017792
	kernel = <<<512, 512>>>, time = 0.029632
	kernel = <<<512, 1024>>>, time = 0.083584
	kernel = <<<1024, 32>>>, time = 0.020384
	kernel = <<<1024, 64>>>, time = 0.020608
	kernel = <<<1024, 128>>>, time = 0.021216
	kernel = <<<1024, 256>>>, time = 0.027776
	kernel = <<<1024, 512>>>, time = 0.050144
	kernel = <<<1024, 1024>>>, time = 0.159360

100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73
74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99
100 99 98 97 96 95 94 93 92 91 90 89 88 87
86 85 84 83 82 81 80 79 78 77 76 75 74 73
72 71 70 69 68 67 66 65 64 63 62 61 60 59
58 57 56 55 54 53 52 51 50 49 48 47 46 45
44 43 42 41 40 39 38 37 36 35 34 33 32 31
30 29 28 27 26 25 24 23 22 21 20 19 18 17
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	kernel = <<<1, 32>>>, time = 0.027200
	kernel = <<<1, 64>>>, time = 0.011232
	kernel = <<<1, 128>>>, time = 0.009792
	kernel = <<<1, 256>>>, time = 0.008960
	kernel = <<<1, 512>>>, time = 0.008960
	kernel = <<<1, 1024>>>, time = 0.008864
	kernel = <<<2, 32>>>, time = 0.009216
	kernel = <<<2, 64>>>, time = 0.009056
	kernel = <<<2, 128>>>, time = 0.009248
	kernel = <<<2, 256>>>, time = 0.008640
	kernel = <<<2, 512>>>, time = 0.008736
	kernel = <<<2, 1024>>>, time = 0.008640
	kernel = <<<4, 32>>>, time = 0.008768
	kernel = <<<4, 64>>>, time = 0.008288
	kernel = <<<4, 128>>>, time = 0.009152
	kernel = <<<4, 256>>>, time = 0.008448
	kernel = <<<4, 512>>>, time = 0.008704
	kernel = <<<4, 1024>>>, time = 0.008800
	kernel = <<<8, 32>>>, time = 0.008480
	kernel = <<<8, 64>>>, time = 0.008704
	kernel = <<<8, 128>>>, time = 0.008928
	kernel = <<<8, 256>>>, time = 0.008896
	kernel = <<<8, 512>>>, time = 0.008768
	kernel = <<<8, 1024>>>, time = 0.009344
	kernel = <<<16, 32>>>, time = 0.008672
	kernel = <<<16, 64>>>, time = 0.008576
	kernel = <<<16, 128>>>, time = 0.009056
	kernel = <<<16, 256>>>, time = 0.008672
	kernel = <<<16, 512>>>, time = 0.008800
	kernel = <<<16, 1024>>>, time = 0.010624
	kernel = <<<32, 32>>>, time = 0.008480
	kernel = <<<32, 64>>>, time = 0.008672
	kernel = <<<32, 128>>>, time = 0.009184

	kernel = <<<32, 256>>>, time = 0.008864
	kernel = <<<32, 512>>>, time = 0.009408
	kernel = <<<32, 1024>>>, time = 0.012800
	kernel = <<<64, 32>>>, time = 0.008800
	kernel = <<<64, 64>>>, time = 0.008832
	kernel = <<<64, 128>>>, time = 0.009408
	kernel = <<<64, 256>>>, time = 0.009376
	kernel = <<<64, 512>>>, time = 0.011008
	kernel = <<<64, 1024>>>, time = 0.017696
	kernel = <<<128, 32>>>, time = 0.009728
	kernel = <<<128, 64>>>, time = 0.010080
	kernel = <<<128, 128>>>, time = 0.013248
	kernel = <<<128, 256>>>, time = 0.010816
	kernel = <<<128, 512>>>, time = 0.013408
	kernel = <<<128, 1024>>>, time = 0.026432
	kernel = <<<256, 32>>>, time = 0.011040
	kernel = <<<256, 64>>>, time = 0.011392
	kernel = <<<256, 128>>>, time = 0.011264
	kernel = <<<256, 256>>>, time = 0.012768
	kernel = <<<256, 512>>>, time = 0.018528
	kernel = <<<256, 1024>>>, time = 0.046080
	kernel = <<<512, 32>>>, time = 0.014176
	kernel = <<<512, 64>>>, time = 0.014400
	kernel = <<<512, 128>>>, time = 0.014400
	kernel = <<<512, 256>>>, time = 0.017792
	kernel = <<<512, 512>>>, time = 0.029600
	kernel = <<<512, 1024>>>, time = 0.083168
	kernel = <<<1024, 32>>>, time = 0.020416
	kernel = <<<1024, 64>>>, time = 0.020832
	kernel = <<<1024, 128>>>, time = 0.021024
	kernel = <<<1024, 256>>>, time = 0.027712
	kernel = <<<1024, 512>>>, time = 0.049728
	kernel = <<<1024, 1024>>>, time = 0.158560

10000
(натуральные числа, одинаковые массивы)
	kernel = <<<1, 32>>>, time = 0.224896
	kernel = <<<1, 64>>>, time = 0.074848
	kernel = <<<1, 128>>>, time = 0.043872
	kernel = <<<1, 256>>>, time = 0.028608
	kernel = <<<1, 512>>>, time = 0.020320
	kernel = <<<1, 1024>>>, time = 0.020224
	kernel = <<<2, 32>>>, time = 0.070944
	kernel = <<<2, 64>>>, time = 0.042144
	kernel = <<<2, 128>>>, time = 0.025568
	kernel = <<<2, 256>>>, time = 0.019136
	kernel = <<<2, 512>>>, time = 0.014528
	kernel = <<<2, 1024>>>, time = 0.015936
	kernel = <<<4, 32>>>, time = 0.039584
	kernel = <<<4, 64>>>, time = 0.026336
	kernel = <<<4, 128>>>, time = 0.018368
	kernel = <<<4, 256>>>, time = 0.015328
	kernel = <<<4, 512>>>, time = 0.014240
	kernel = <<<4, 1024>>>, time = 0.015520
	kernel = <<<8, 32>>>, time = 0.025664
	kernel = <<<8, 64>>>, time = 0.018048
	kernel = <<<8, 128>>>, time = 0.015712
	kernel = <<<8, 256>>>, time = 0.013888
	kernel = <<<8, 512>>>, time = 0.013824

	kernel = <<<8, 1024>>>, time = 0.015936
	kernel = <<<16, 32>>>, time = 0.018752
	kernel = <<<16, 64>>>, time = 0.015136
	kernel = <<<16, 128>>>, time = 0.014240
	kernel = <<<16, 256>>>, time = 0.014432
	kernel = <<<16, 512>>>, time = 0.015104
	kernel = <<<16, 1024>>>, time = 0.016320
	kernel = <<<32, 32>>>, time = 0.018848
	kernel = <<<32, 64>>>, time = 0.015392
	kernel = <<<32, 128>>>, time = 0.013440
	kernel = <<<32, 256>>>, time = 0.014144
	kernel = <<<32, 512>>>, time = 0.015104
	kernel = <<<32, 1024>>>, time = 0.018944
	kernel = <<<64, 32>>>, time = 0.017472
	kernel = <<<64, 64>>>, time = 0.015328
	kernel = <<<64, 128>>>, time = 0.015840
	kernel = <<<64, 256>>>, time = 0.012352
	kernel = <<<64, 512>>>, time = 0.014112
	kernel = <<<64, 1024>>>, time = 0.021280
	kernel = <<<128, 32>>>, time = 0.015968
	kernel = <<<128, 64>>>, time = 0.014560
	kernel = <<<128, 128>>>, time = 0.013760
	kernel = <<<128, 256>>>, time = 0.015552
	kernel = <<<128, 512>>>, time = 0.018816
	kernel = <<<128, 1024>>>, time = 0.032960
	kernel = <<<256, 32>>>, time = 0.021504
	kernel = <<<256, 64>>>, time = 0.019040
	kernel = <<<256, 128>>>, time = 0.018400
	kernel = <<<256, 256>>>, time = 0.020224
	kernel = <<<256, 512>>>, time = 0.025728
	kernel = <<<256, 1024>>>, time = 0.050304
	kernel = <<<512, 32>>>, time = 0.020032
	kernel = <<<512, 64>>>, time = 0.017856
	kernel = <<<512, 128>>>, time = 0.017216
	kernel = <<<512, 256>>>, time = 0.020448
	kernel = <<<512, 512>>>, time = 0.032608
	kernel = <<<512, 1024>>>, time = 0.087136
	kernel = <<<1024, 32>>>, time = 0.026176
	kernel = <<<1024, 64>>>, time = 0.023968
	kernel = <<<1024, 128>>>, time = 0.023936
	kernel = <<<1024, 256>>>, time = 0.030304
	kernel = <<<1024, 512>>>, time = 0.053824
	kernel = <<<1024, 1024>>>, time = 0.161696

100000	kernel = <<<1, 32>>>, time = 2.077984
	kernel = <<<1, 64>>>, time = 1.054688
	kernel = <<<1, 128>>>, time = 0.540160
	kernel = <<<1, 256>>>, time = 0.278560
	kernel = <<<1, 512>>>, time = 0.156224
	kernel = <<<1, 1024>>>, time = 0.128448
	kernel = <<<2, 32>>>, time = 1.041536
	kernel = <<<2, 64>>>, time = 0.537344
	kernel = <<<2, 128>>>, time = 0.277376
	kernel = <<<2, 256>>>, time = 0.152992
	kernel = <<<2, 512>>>, time = 0.094208
	kernel = <<<2, 1024>>>, time = 0.086720
	kernel = <<<4, 32>>>, time = 0.537952

	kernel = <<<4, 64>>>, time = 0.276864
	kernel = <<<4, 128>>>, time = 0.153664
	kernel = <<<4, 256>>>, time = 0.093952
	kernel = <<<4, 512>>>, time = 0.090176
	kernel = <<<4, 1024>>>, time = 0.094368
	kernel = <<<8, 32>>>, time = 0.277184
	kernel = <<<8, 64>>>, time = 0.154080
	kernel = <<<8, 128>>>, time = 0.094272
	kernel = <<<8, 256>>>, time = 0.081568
	kernel = <<<8, 512>>>, time = 0.082592
	kernel = <<<8, 1024>>>, time = 0.083168
	kernel = <<<16, 32>>>, time = 0.152832
	kernel = <<<16, 64>>>, time = 0.096416
	kernel = <<<16, 128>>>, time = 0.082720
	kernel = <<<16, 256>>>, time = 0.083552
	kernel = <<<16, 512>>>, time = 0.082560
	kernel = <<<16, 1024>>>, time = 0.083648
	kernel = <<<32, 32>>>, time = 0.151360
	kernel = <<<32, 64>>>, time = 0.097408
	kernel = <<<32, 128>>>, time = 0.080992
	kernel = <<<32, 256>>>, time = 0.081344
	kernel = <<<32, 512>>>, time = 0.080032
	kernel = <<<32, 1024>>>, time = 0.083360
	kernel = <<<64, 32>>>, time = 0.117408
	kernel = <<<64, 64>>>, time = 0.084736
	kernel = <<<64, 128>>>, time = 0.079616
	kernel = <<<64, 256>>>, time = 0.079968
	kernel = <<<64, 512>>>, time = 0.079040
	kernel = <<<64, 1024>>>, time = 0.086240
	kernel = <<<128, 32>>>, time = 0.118400
	kernel = <<<128, 64>>>, time = 0.084896
	kernel = <<<128, 128>>>, time = 0.078880
	kernel = <<<128, 256>>>, time = 0.078304
	kernel = <<<128, 512>>>, time = 0.078752
	kernel = <<<128, 1024>>>, time = 0.093248
	kernel = <<<256, 32>>>, time = 0.113504
	kernel = <<<256, 64>>>, time = 0.081984
	kernel = <<<256, 128>>>, time = 0.078336
	kernel = <<<256, 256>>>, time = 0.077952
	kernel = <<<256, 512>>>, time = 0.080032
	kernel = <<<256, 1024>>>, time = 0.112512
	kernel = <<<512, 32>>>, time = 0.115584
	kernel = <<<512, 64>>>, time = 0.083776
	kernel = <<<512, 128>>>, time = 0.077408
	kernel = <<<512, 256>>>, time = 0.079008
	kernel = <<<512, 512>>>, time = 0.090528
	kernel = <<<512, 1024>>>, time = 0.150176
	kernel = <<<1024, 32>>>, time = 0.115488
	kernel = <<<1024, 64>>>, time = 0.083712
	kernel = <<<1024, 128>>>, time = 0.080288
	kernel = <<<1024, 256>>>, time = 0.088704
	kernel = <<<1024, 512>>>, time = 0.111712
	kernel = <<<1024, 1024>>>, time = 0.225600

1000000	kernel = <<<1, 32>>>, time = 20.691263
	kernel = <<<1, 64>>>, time = 10.530560
	kernel = <<<1, 128>>>, time = 5.254464

	kernel = <<<1, 256>>>, time = 2.712832
	kernel = <<<1, 512>>>, time = 1.488928
	kernel = <<<1, 1024>>>, time = 1.148160
	kernel = <<<2, 32>>>, time = 10.379840
	kernel = <<<2, 64>>>, time = 5.257760
	kernel = <<<2, 128>>>, time = 2.683872
	kernel = <<<2, 256>>>, time = 1.475136
	kernel = <<<2, 512>>>, time = 0.879488
	kernel = <<<2, 1024>>>, time = 0.783264
	kernel = <<<4, 32>>>, time = 5.243488
	kernel = <<<4, 64>>>, time = 2.698880
	kernel = <<<4, 128>>>, time = 1.472704
	kernel = <<<4, 256>>>, time = 0.878464
	kernel = <<<4, 512>>>, time = 0.812672
	kernel = <<<4, 1024>>>, time = 0.819040
	kernel = <<<8, 32>>>, time = 2.700800
	kernel = <<<8, 64>>>, time = 1.474880
	kernel = <<<8, 128>>>, time = 0.884928
	kernel = <<<8, 256>>>, time = 0.729952
	kernel = <<<8, 512>>>, time = 0.727808
	kernel = <<<8, 1024>>>, time = 0.730944
	kernel = <<<16, 32>>>, time = 1.476224
	kernel = <<<16, 64>>>, time = 0.875648
	kernel = <<<16, 128>>>, time = 0.727072
	kernel = <<<16, 256>>>, time = 0.739360
	kernel = <<<16, 512>>>, time = 0.738432
	kernel = <<<16, 1024>>>, time = 0.742208
	kernel = <<<32, 32>>>, time = 1.450624
	kernel = <<<32, 64>>>, time = 0.949856
	kernel = <<<32, 128>>>, time = 0.756256
	kernel = <<<32, 256>>>, time = 0.715104
	kernel = <<<32, 512>>>, time = 0.706624
	kernel = <<<32, 1024>>>, time = 0.717632
	kernel = <<<64, 32>>>, time = 1.118240
	kernel = <<<64, 64>>>, time = 0.841408
	kernel = <<<64, 128>>>, time = 0.718464
	kernel = <<<64, 256>>>, time = 0.712896
	kernel = <<<64, 512>>>, time = 0.708672
	kernel = <<<64, 1024>>>, time = 0.716576
	kernel = <<<128, 32>>>, time = 1.201824
	kernel = <<<128, 64>>>, time = 0.780000
	kernel = <<<128, 128>>>, time = 0.708128
	kernel = <<<128, 256>>>, time = 0.704640
	kernel = <<<128, 512>>>, time = 0.700256
	kernel = <<<128, 1024>>>, time = 0.730720
	kernel = <<<256, 32>>>, time = 1.156832
	kernel = <<<256, 64>>>, time = 0.733120
	kernel = <<<256, 128>>>, time = 0.703200
	kernel = <<<256, 256>>>, time = 0.701888
	kernel = <<<256, 512>>>, time = 0.699808
	kernel = <<<256, 1024>>>, time = 0.736352
	kernel = <<<512, 32>>>, time = 1.063712
	kernel = <<<512, 64>>>, time = 0.740096
	kernel = <<<512, 128>>>, time = 0.699168
	kernel = <<<512, 256>>>, time = 0.698464
	kernel = <<<512, 512>>>, time = 0.697088
	kernel = <<<512, 1024>>>, time = 0.748768

	kernel = <<<1024, 32>>>, time = 1.118656
	kernel = <<<1024, 64>>>, time = 0.727456
	kernel = <<<1024, 128>>>, time = 0.699168
	kernel = <<<1024, 256>>>, time = 0.696064
	kernel = <<<1024, 512>>>, time = 0.696608
	kernel = <<<1024, 1024>>>, time = 0.836640

10000000	kernel = <<<1, 32>>>, time = 206.393219
	kernel = <<<1, 64>>>, time = 105.431999
	kernel = <<<1, 128>>>, time = 52.587486
	kernel = <<<1, 256>>>, time = 27.024160
	kernel = <<<1, 512>>>, time = 14.637216
	kernel = <<<1, 1024>>>, time = 11.513984
	kernel = <<<2, 32>>>, time = 104.625923
	kernel = <<<2, 64>>>, time = 52.489056
	kernel = <<<2, 128>>>, time = 26.853567
	kernel = <<<2, 256>>>, time = 14.540448
	kernel = <<<2, 512>>>, time = 8.693216
	kernel = <<<2, 1024>>>, time = 7.669792
	kernel = <<<4, 32>>>, time = 52.891422
	kernel = <<<4, 64>>>, time = 26.979551
	kernel = <<<4, 128>>>, time = 14.532992
	kernel = <<<4, 256>>>, time = 8.678592
	kernel = <<<4, 512>>>, time = 8.086592
	kernel = <<<4, 1024>>>, time = 8.104352
	kernel = <<<8, 32>>>, time = 26.963680
	kernel = <<<8, 64>>>, time = 14.518112
	kernel = <<<8, 128>>>, time = 8.613312
	kernel = <<<8, 256>>>, time = 7.252800
	kernel = <<<8, 512>>>, time = 7.246560
	kernel = <<<8, 1024>>>, time = 7.154112
	kernel = <<<16, 32>>>, time = 14.480000
	kernel = <<<16, 64>>>, time = 8.597280
	kernel = <<<16, 128>>>, time = 7.252800
	kernel = <<<16, 256>>>, time = 7.310112
	kernel = <<<16, 512>>>, time = 7.302720
	kernel = <<<16, 1024>>>, time = 7.278496
	kernel = <<<32, 32>>>, time = 14.400352
	kernel = <<<32, 64>>>, time = 9.688256
	kernel = <<<32, 128>>>, time = 7.512960
	kernel = <<<32, 256>>>, time = 7.120128
	kernel = <<<32, 512>>>, time = 7.023456
	kernel = <<<32, 1024>>>, time = 7.024768
	kernel = <<<64, 32>>>, time = 11.301792
	kernel = <<<64, 64>>>, time = 8.282496
	kernel = <<<64, 128>>>, time = 7.156096
	kernel = <<<64, 256>>>, time = 7.068064
	kernel = <<<64, 512>>>, time = 7.030624
	kernel = <<<64, 1024>>>, time = 7.037024
	kernel = <<<128, 32>>>, time = 12.262656
	kernel = <<<128, 64>>>, time = 7.743232
	kernel = <<<128, 128>>>, time = 7.022464
	kernel = <<<128, 256>>>, time = 6.994752
	kernel = <<<128, 512>>>, time = 6.959456
	kernel = <<<128, 1024>>>, time = 7.019712
	kernel = <<<256, 32>>>, time = 11.622176

	kernel = <<<256, 64>>>, time = 7.289984
	kernel = <<<256, 128>>>, time = 7.007136
	kernel = <<<256, 256>>>, time = 6.968160
	kernel = <<<256, 512>>>, time = 6.956800
	kernel = <<<256, 1024>>>, time = 7.017312
	kernel = <<<512, 32>>>, time = 10.614240
	kernel = <<<512, 64>>>, time = 7.366944
	kernel = <<<512, 128>>>, time = 6.979104
	kernel = <<<512, 256>>>, time = 6.957280
	kernel = <<<512, 512>>>, time = 6.947520
	kernel = <<<512, 1024>>>, time = 7.040928
	kernel = <<<1024, 32>>>, time = 11.458048
	kernel = <<<1024, 64>>>, time = 7.212864
	kernel = <<<1024, 128>>>, time = 6.966656
	kernel = <<<1024, 256>>>, time = 6.957728
	kernel = <<<1024, 512>>>, time = 6.928064
	kernel = <<<1024, 1024>>>, time = 7.085184

Работа на CPU:

Тест:	Результат:

3
1 2 3
4 5 6
	0.002

100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	0.004

10000 (натуральные числа, одинаковые массивы)	**0.059**
100000	**0.567**
1000000	**5.452**

10000000	**33.986**

Код программы для CPU:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main() {
    long long n;
    scanf("%lld", &n);

    double* v1 = (double*)malloc(n * sizeof(double));
    double* v2 = (double*)malloc(n * sizeof(double));

    for(long long i = 0; i < n; i++) {
        v1[i] = i;
    }
    for(long long i = 0; i < n; i++) {
        v2[i] = n - i;
   }

    clock_t begin = clock();
    for (long long i =0; i < n; i++) {
        if (v1[i] > v2[i]) {
            v1[i] = v2[i];
        }
    }
    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("%lf\n", time_spent);
    printf("\n");
    free(v1);
    free(v2);
    return 0;
}
```

Выводы

Программирование с распараллеливанием на CUDA осмысленно для больших объемов данных. На представленных мной маленьких массивах такое количество вызванных потоков в kernel лишь замедлило программу, потому что были “лишние” вычисления по определению, какому потоку что делать, и программа чисто на CPU справилась гораздо быстрее, просто последовательно сравнивая все элементы. Поэтому увидеть эффективность программы для GPU можно на очень больших данных, больше порядка 1000000, там время уже на порядок отличается.

Единственная сложность заключается в понимании, что GPU не умеет работать с данными записанными на CPU. Также важно понимать, что без написания специального макроса по отслеживанию ошибок, невозможно будет найти, почему программа отработала некорректно.