Лабораторная работа №1

по курсу

«Программирование графических процессоров»

Выполнил:

Смирнов Александр

Группа 21225

Задание

Заполнить на графическом процессоре массив типа float/double значениями синуса (один период на всю длину массива). Размер массива - 10^7 элементов. Для заполненного массива на графическом процессоре посчитать сумму всех элементов массива. Сравнить со значением, вычисленном на центральном процессоре. Сравнить разультат для массивов  float и double.

Решение

Исходный код расположен в репозитории <https://github.com/SmirAlex/gpu-programming> в папке lab1.

Описание решения

Были разработаны 2 версии программы: одна для CPU, другая – для GPU. Названия файлов соответствующие. Для измерения времени в обоих случаях была использована функция clock\_gettime из time.h, которая предоставляет дает время с точностью до наносекунд. Для каждого случая измерения проводились 5 раз, в результате выводилось среднее значение по итогу 5 испытаний. Тип элементов массива (float или double) задается через соответствующий идентификатор с помощью директивы #define. Результаты измерений приведены в таблице ниже.

|  |  |  |
| --- | --- | --- |
|  | CPU (sum result / time) | GPU (sum result / time) |
| float | -0.04986 / 295.07 ms | 0.01639 / 33.46 ms |
| double | -0.0 / 301.01 ms | 0.0 / 35.72 ms |

Из измерений можно сделать вывод, что вычисления на GPU и точнее, и значительно быстрее, чем на CPU. Как на CPU, так и на GPU точность зависит от используемого типа данных с плавающей точкой: float не обеспечивает необходимой точности для многочисленных операций сложения и умножения вещественных чисел, отсюда и такая большая погрешность относительно правильного значения (нуля), точности типа double же достаточно и для CPU, и для GPU.

В программе на GPU использовались следующие директивы:

* #pragma acc data create(sin\_arr[0:N])
  + Выделение памяти на GPU без копирования
* #pragma acc kernels
  + Компилятор самостоятельно решает как параллелить вычисления
* #pragma acc kernels loop reduction(+:sum\_sin)
  + Параллельные вычисления + цикл с редукцией по суммированию

Вывод профилирования программы после PGI\_ACC\_TIME=1

compute\_sum\_sin\_gpu NVIDIA devicenum=0

time(us): 2,703

21: compute region reached 1 time

25: kernel launched 1 time

grid: [65535] block: [128]

device time(us): total=1,869 max=1,869 min=1,869 avg=1,869

elapsed time(us): total=1,917 max=1,917 min=1,917 avg=1,917

21: data region reached 2 times

27: compute region reached 1 time

31: kernel launched 1 time

grid: [65535] block: [128]

device time(us): total=677 max=677 min=677 avg=677

elapsed time(us): total=706 max=706 min=706 avg=706

31: reduction kernel launched 1 time

grid: [1] block: [256]

device time(us): total=111 max=111 min=111 avg=111

elapsed time(us): total=130 max=130 min=130 avg=130

27: data region reached 2 times

27: data copyin transfers: 1

device time(us): total=10 max=10 min=10 avg=10

33: data copyout transfers: 1

device time(us): total=36 max=36 min=36 avg=36

Отсюда видно, что ядро было запущено 3 раза: 2 раза на каждый цикл + 1 раз для выполнения редукции. Дело в том, что параллельное суммирование работает в 2 этапа: сначала массив разбивается на блоки, для каждого из которого считается сумма, а затем полученные суммы складываются (происходит редукция).