МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу «Программирование графических процессоров»

Классификация и кластеризация изображений на GPU.

Выполнил: А. Е. Максимов
Группа: М8О-407Б-19
Преподаватель: А. Ю. Морозов

Москва, 2023

1 Условие

Цель работы:Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков.

Вариант на "два". Трех-цветовой классификатор.

Необходимо реализовать вариант №3 при фиксированных параметрах:

$nc = 3$

$avg_0 = (255, 0, 0)^T$

$avg_1 = (0, 255, 0)^T$

$avg_2 = (0, 0, 255)^T$

Ограничения:

- $w * h \leq 4 * 10^8$

2 Программное и аппаратное обеспечение

Характеристики графического процессора:

- Compute capability: 7.5
- Наименование: NVIDIA GeForce GTX 1660 Ti
- Графическая память: 6143mb
- Константная память: 64kb
- Разделяемая память: 48kb
- Количество регистров на блок: 65536
- Максимальное количество потоков на блок: (1024, 1024, 64)
- Максимальное количество блоков: (2147483647, 65535, 65535)
- Количество мультипроцессоров: 6

Характеристики системы:

- Процессор: «Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz»
- Память: 16,0 ГБ встроенной памяти ноутбука
- SSD: «Hynix BC511 NVMe SK hynix 512GB »

Программное обеспечение:

- ОС: «Windows 10 Домашняя x64»
- Текстовый редактор: «Notepad++»
- Компилятор: «nvcc: Cuda compilation tools, release 12.0, V12.0.76»

3 Метод решения

Для выполнения этой лабораторной работы я использовал статический массив классов, который передавался в константную память графического процессора, после чего использовался для простой классификации пикселей входных данных.

Для небольшой оптимизации работы варпов я решил заменить условные операторы $if()$ на логические переменные, отвечающие вопросу больше или меньше расстояние от текущего пикселя картины до характеристического пикселя класса. Принципиально это не было большим улучшением, однако позволило тренировать мозг на оптимальное написание кода для графических процессоров.

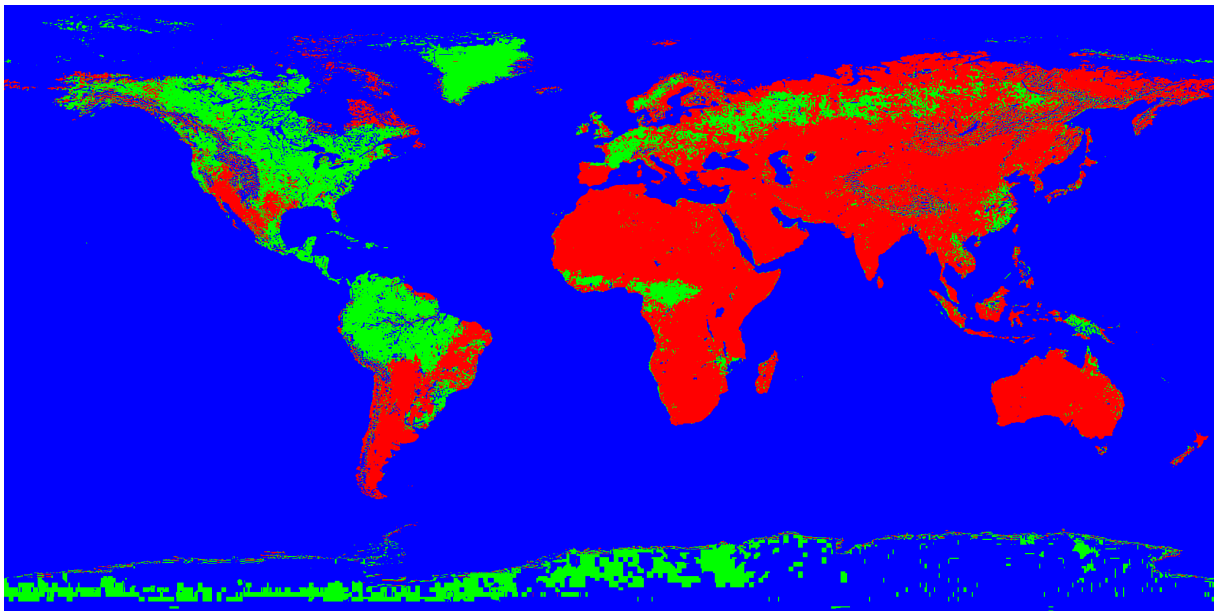
4 Результаты

Замеры времени работы CPU и ядер с различными конфигурациями.

Данные представлены квадратными таблицами размера $n * n$.

Конфигурация	$n = 10$, мс	$n = 100$, мс	$n = 10^3$, мс	$n = 10^4$, мс
CPU	0.0	0.0	23.973	2355.389
(1, 32)	0.0051	0.0675	10.486	902.33
(1, 256)	0.0058	0.0143	1.39	136.626
(1, 1024)	0.0055	0.0087	0.4578	45.138
(32, 32)	0.0049	0.0062	0.3463	34.0766
(32, 256)	0.0042	0.0061	0.0676	6.6293
(32, 1024)	0.0041	0.0056	0.0489	5.1194
(64, 32)	0.0044	0.0060	0.1822	18.2907
(64, 256)	0.0054	0.0057	0.0424	4.9502
(64, 1024)	0.0048	0.0045	0.0406	4.2377
(256, 32)	0.0059	0.0056	0.0671	6.5016
(256, 64)	0.0044	0.0061	0.0430	5.0606
(256, 256)	0.0045	0.0053	0.0379	4.2345

Для демонстрации работы программы я использовал предоставленный пример с курса.



5 Выводы

Константная память является удобным контейнером для хранения часто используемых в нитях данных. В задачах классификации она черезвычайно полезна, поскольку отсутствует необходимость записи новых данных и требуется множество обращений от разных нитей и блоков.

Сложность алгоритма $O(w * h)$.

Список литературы

[1] *CUDA C++ Programming Guide*
URL: `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#`