Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Практикум на ЭВМ: 7 семестр.

Отчёт № 1.

Анализ программы на CUDA, реализующей свертку изображения

Работу выполнил

Федоров В. В.

Москва 2021

Постановка задачи и формат данных.

Задача: Реализовать программу с использованием CUDA, осуществляющей свертку изображения с тремя заранее выбранными ядрами (2 ядра 3x3 и 1 ядро 5x5), протестировать программу в двух режимах – на большом изображении размером минимум 2000x2000 и на множестве малых изображений размером порядка 300x300.

Математическое описание

В общем виде свертка является операцией над произвольной матрицей A размеров $N \times M$ и другой матрицей K размера $2p+1 \times 2q+1$, называемой ядром свертки. Выходом этой операции является матрица B, вычисляемая по следующей формуле:

$$B[x][y] = \sum_{i=-p}^{p} \sum_{j=-q}^{q} A[x+i][y+j] \times K[i][j]$$

Для простоты элементы ядра нумеруем, начиная с $(-p, -q)$. Для того, чтобы не выходить за границу входного массива, размер выходного массива получается равным $N-2p \times M-2q$. В случае изображения оно представляется в виде нескольких матриц, каждая из которых сворачивается по отдельности. Для тестирования программы были выбраны следующие 3 ядра:

Увеличение резкости (Sharpen): $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Обнаружение границ (Edge detection): $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Гауссово размытие (Gauss blur): $\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$

Примеры работы свертки с данными ядрами:

Оригинал	Sharpen	Edge detection	Gauss blur
			

Описание программы

Программа состоит из следующих элементов:

- `__global__ void conv(Pixel *src, Pixel *tgt, double *ker, int ker_pad, int width, int height);` - функция, выполняемая CUDA-ядрами. Осуществляет свертку для одного пикселя выходного изображения.
- `class CudaProcessor;` - класс, хранящий в себе выделенные в оперативной памяти GPU массивы для входного и выходного изображения и для ядра, а также некоторые числовые параметры вроде размеров изображений и замеров по времени. Он имеет следующие методы:

    - `CudaProcessor(int width, int height, double* convker, int _ker_dim);` - выделяет необходимую память на GPU.
    - `void ProcessImage(Image& input, Image& output);` - обрабатывает входное изображение и сохраняет результат в выходное. Также записывает время работы, затраченное на копирование данных и работу CUDA-ядер.
    - `float KernelTime();` - выводит суммарное время, потраченное на работу CUDA-ядер по всем обработанным изображениям.
    - `float TotallTime();` - выводит суммарное время, потраченное на работу CUDA-ядер и копирование данных по всем обработанным изображениям.
    - `~CudaProcessor();` - освобождает выделенную память.
- `int main(int argc, char *argv[]);` - основной цикл работы программы, открывает изображения, обрабатывает их и сохраняет, после чего выводит два замера времени.

Программа написана с использованием CMake и библиотеки STB для открытия и сохранения изображений. После сборки из папки bin необходимо запустить программу при помощи команды `./main <sharpen/edge/gauss5> <large/small>`
В случае аргумента large программа обрабатывает 1 изображение размером 5000x5000, в случае small – 1000 изображений размером 300x300.

Тестирование и результаты

Программа запускалась на системе Polus. Тестирование показало следующие результаты по времени:

Время работы ядер, с

	Large	Small
Sharpen	0.007007	0.045545
Edge	0.007051	0.046260
Gauss5	0.016888	0.082372

Время работы ядер и копирований, с

	Large	Small
Sharpen	0.049590	0.137777
Edge	0.050056	0.135577
Gauss5	0.060111	0.169104

Основные выводы.

Программа тратит на порядок больше времени на копирование данных чем на их обработку на GPU. Помимо этого, в режиме Small намного больше времени тратится на инициализацию копирований и вызовов ядра в силу того, что они запускаются 1000 раз.