ИССЛЕДОВАНИЕ ПОДХОДОВ К РЕАЛИЗАЦИИ ОПЕРАЦИИ СВЁРТКИ НА ОСНОВЕ ТЕХНОЛОГИИ CUDA

Хурсов П. С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Перцев Д. Ю. – ст. преп.

Одной из наиболее распространенных операций над изображениями является фильтрация на основе операции свертки. Для решения данной задачи используются графические ускорители с поддержкой технологии CUDA. В данной работе приводится обзор различных подходов к реализации операции свертки.

Наиболее распространенным форматом изображений является RGB888, при котором каждый пиксель изображения кодируется 3 байтами, описывающие интенсивность красной, зеленой и синей компоненты. При таком формате изображение зачастую хранится в памяти в виде плоского массива размерностью NxMx3 байта, где N – высота, M – ширина изображения.

Для эффективной работы с памятью в CUDA необходимо следовать некоторым правилам, в частности, формировать транзакции при работе с глобальной памятью. Для этого необходимо, чтобы адреса памяти, к которым обращается warp, были выравнены на границу 128 байт. При работе с одномерным массивом пикселей данное условие легко нарушить, так как адрес начала некоторой строки зависит от ширины изображения. Чтобы исправить данный недостаток используется:

– функция cudaMallocPitch, которая выделяет расширенный объем память для матрицы NxM, добавляя некоторое количество памяти в конец каждой строки, чтобы сделать адрес начала каждой

строки гарантированно кратным 128;

– cudaMemcpy2D, которая позволяет эффективно работать с выравненной памятью.

Альтернативным подходом является использование текстурной памяти [1]. Однако данный тип памяти может работать только с 1-, 2- и 4-байтными словами. Для работы с текстурной памятью необходимо ввести альфа-канал, что увеличит расход памяти и потребует некоторой предобработки изображения. Сравнение скорости работы операции свертки над выравненной и не выравненной матрицами приведен на рис. 1. Так как основным узким местом на данном этапе является обращение к исходному изображению, сравнение ведется по размеру изображения при фиксированной ширине, что наглядно демонстрирует важность выравнивания начала строк в памяти.

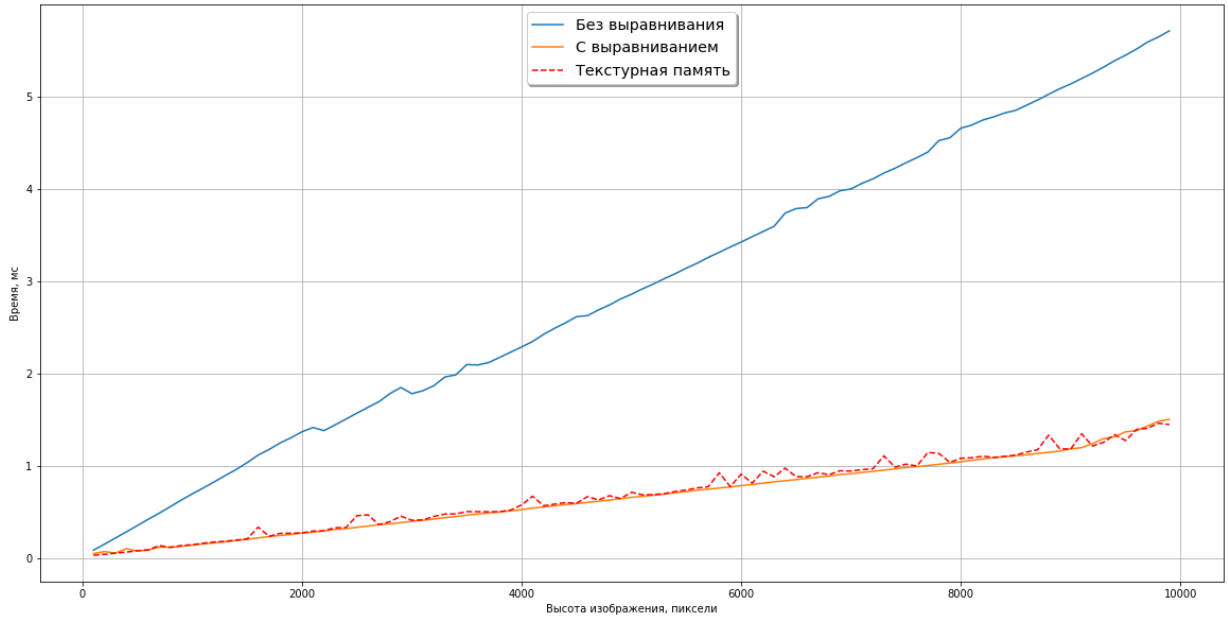

Рисунок 1 – Сравнения времени обработки изображения различными способами работы с памятью

Как видно, наилучшими способами работой с памятью являются выравненная и текстурная память. Также видно, что по времени они не сильно отличаются и между ними следует выбирать исходя из других факторов, например, объема памяти. Также NVIDIA API обязывает использовать выравненную память для инициализации текстурной, однако текстурная память предоставляет более удобный способ работы с границами изображения.

Другим условием формирования 128-байтной транзакции является обращение к 4-байтным словам. Однако, при использовании глобальной памяти каждый пиксель задается 3 байтами, т. е. при подходе одна нить – один пиксель формирования транзакции не удастся достичь. Если посмотреть на расположение изображения в памяти (рис. 2), видно, что при подходе одна нить – четыре пикселя возможно достигнуть формирования трех 128-байтных транзакций.

Рисунок 2 – Расположение изображения в памяти.

Свертка задается выражением:

$$D[i,j] = \sum_{m,n} S[i-m, j-n] * K[m,n] \quad (1)$$

где D – матрица отфильтрованного изображения, S – матрица исходного изображения, K – ядро свертки, i, j – координаты пикселя.

При вычислении соседних пикселей значения из матрицы исходного изображения и ядра свертки используются по несколько раз. Технология CUDA предоставляет в распоряжение

программисту разделяемую память. Доступ к разделяемой памяти для потоков в пределах одного блока исполняемого ядра CUDA гораздо быстрее, чем доступ к глобальной или текстурной памяти. Используя это можно существенно ускорить выполнение операции свертки.

Время выполнения также зависит от типа арифметики. Основными вариантами являются целочисленная и арифметика с одинарной точностью (арифметика с двойной точностью в данной задаче имеет мало смысла, так как формат изображения целочисленный и нет необходимости в повышенной точности). Сравнение различных вариантов реализации ядра с использованием разделяемой памяти и без, а также различных типов данных показан на рис. 3.

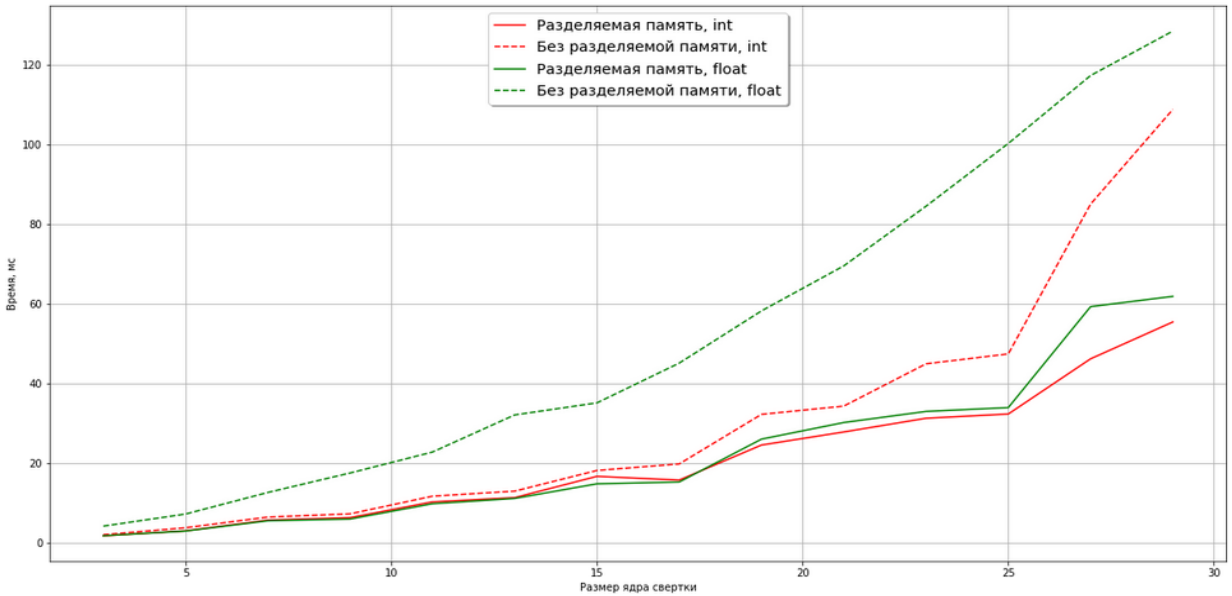

Рисунок 3 – Зависимость времени выполнения операции свертки от размера ядра матрицы свертки.

Как видно из графиков оптимальным с точки зрения скорости исполнения является использование разделяемой памяти с целочисленной арифметикой. Таким образом, оптимальной с точки зрения использования памяти и быстродействия является реализация, использующая выравненную глобальную память, разделяемую память внутри блока и целочисленную арифметику.

Список использованных источников:

1. Основы работы с технологией CUDA /Боресков А. В., Харламов А. А. // М.: ДМК Пресс, 2010. – С. 129-131.