УЛУЧШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ОПРЕДЕЛЕНИЯ ГРАНИЦ СЦЕН ВИДЕО С ИСПОЛЬЗОВАНИЕМ OPENCL

Потапьев И.А.
Научный руководитель: Аксенов С.В., к.т.н., доцент
Национальный исследовательский Томский политехнический университет
E-mail: ipotapev@gmail.com

Данная статья посвящена реализации алгоритма обработки видео на OpenCL. В статье описаны некоторые техники и методы, позволяющие получить прирост производительности разработанной системы.

1. Введение

Специфика задач компьютерного зрения предполагает использование вычислительно сложных операций. Многие из этих задачи включают в себя работу непосредственно с изображением, в частности матричные операции.

Так, алгоритм может содержать в себе большое количество последовательных операций, где результат выполнения предыдущих не влияет на вычисление последующих. Это значит, что данный тип алгоритмов может быть распараллелен. Выполнение же его в последовательном виде будет весьма медленным и неэффективным. Цель данной работы – реализовать параллельную версию алгоритма.

Изначальная система представляет собой библиотеку, определяющую границы сцен на данном видеоролике. Эта библиотека может быть использована в широком круге задач, таких как индексация, редактирование видео, а также видеоэффекты.

Таким образом, возникла задача выбора технологии параллельных вычислений, которые подходили бы как для серверных решений, так и для десктопов. Большинство из рассмотренных технологий является узкоспециализированными. Например, MPI ориентирована на кластерные вычисления, а OpenMP используется для создания параллельного кода для систем симметричного мультипроцессирования. В конце концов, была выбрана технология OpenCL. Это решение было обусловлено следующими причинами:

- Программа должна работать как на серверах, так и на десктопах
- OpenCL ориентирован на выполнение математических задач
- OpenCL поддерживается основными производителями (nVidia, ATI, Intel , и т.д.)

2. Архитектура

Архитектура системы была значительно пересмотрена и переписана в целях разработки чистого и поддерживаемого параллельного кода. Написание параллельного алгоритма с использованием любой технологии имеет немало сложностей и осуществляется гораздо медленнее, чем написание последовательного алгоритма. Описание оптимизируемой системы определения границ сцены можно найти в [1]. Так, изменение архитектуры привело к следующим свойствам системы.

Первое свойство – OpenCL и C++ версии взаимозаменяемы. Кроме того, обе версии содержатся в одном проекте в целях быстрой сборки и исполнения обоих решений. Библиотека может быть собрана с OpenCL или без нее, в зависимости от настроек.

Другое свойство – наличие в системе специальных программ (*.cl) для GPU, дублирующих функции на C++. Некоторые примеры функций перечислены ниже:

- генерация распределения Гаусса
- 2D свертка
- нормализация цветов изображения
- создание пирамид для изображения.

Прочие аспекты архитектуры:

- сниженное число взаимодействий между оперативной и видеопамятью. копирование изображения происходит дважды: отправка изображения в видеопамять и получение обратно в оперативную память. Все вычисления производятся на GPU
- модульность системы допускает использование кода на GPU фермах.

3. Оптимизация

После того, как разработка системы с описанной выше архитектурой была выполнена, были проведены тесты производительности.

Эти тесты показали большой прирост производительности на видео высокого разрешения, но на видео с низким разрешением прироста не наблюдалось.

В таблице 1 показаны результаты профилирования до оптимизации.

Таблица 1.
Результаты тестирования до оптимизации

Название ядра	***Полное время выполнения (мс)***
filter2D	1519.29566
buildGaussianPyramidSmooth	776.92347
upSize	575.47328
iterativeInteraction	488.98443
naiveSummation	331.11482

Данная таблица содержит наиболее затратные по времени операции. Как видно, большинство

операций было связано со сверткой. Применение фильтров является крайне затратной операцией из-за использования больших ядер (25x25, 21x21). Так, filter2D является наиболее значимым объектом оптимизации. Следующие разделы описывают приемы, увеличивающие скорость выполнения OpenCL кода

3.1 Дивергенция кода

Дивергенция означает, что все потоки имеют приблизительно схожее время выполнения. Если алгоритм программы содержит в себе множество условных ветвлений, то данный алгоритм будет медленней, чем при отсутствии этих ветвлений. В данном случае избавление от ветвлений дало значительный прирост в скорости.

3.2 Улучшенное использование памяти

Глобальная память OpenCL имеет большой объем, низкую скорость доступа. Единственный путь реализации эффективного решения – использовать локальные и константные зоны памяти. Больше информации о модели памяти доступно в [2] и [3].

Буферы памяти можно объявить __constant если буферы используются только для чтения. Так, все буферы фильтров были перенесены из глобальной памяти в константную.

Другой способ повысить скорость операции свертки – избегать чтения соседних пикселей из глобальной памяти [4]. Поэтому ядро filter2D было изменено для использования групп размером 16x16. Элементы одной группы загружают часть изображения в локальную память и затем используют эти данные для операции свертки.

Хорошей практикой является держать размер аргументов для ядра малым настолько, насколько это возможно. Как видно из таблицы 2, filter2D разделен на два компонента. Это позволило не передавать в качестве аргумента размер ядра, а определять его константой. Размер ядра используется в коде цикла, поэтому компилятор может однозначно развернуть цикл [5].

Таблица 2.
Результаты тестирования после оптимизации

Название ядра	***Полное время выполнения (мс)***
filter2DDog	768.55967
filter2DGabor	464.40411
buildGaussianPyramidSmooth	449.49778
upSize	318.26241
iterativeInteraction	312.22124

4. Тестирование

После реализации оптимизированной версии системы были проведены тесты производительности. В таблице 2 представлены результаты по работы ядер OpenCL. Графики времен обработки одного фрейма представлены на рисунке 2. Очевидно, что GPGPU реализация показала себя гораздо более эффективной, чем последовательная на CPU. Тесты проводились на ПК с процессором Intel i5-430m и видеокартой AMD Radeon HD 5850M.

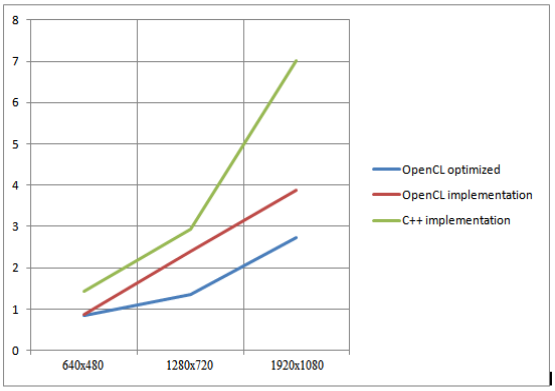

Рис. 1. Времена обработки одного кадра для разных реализаций системы.

5. Заключение

Статья описывает реализацию библиотеки, определяющей границы сцен на GPU, а также набор техник оптимизации, примененных на наиболее часто использующихся функциях приложения, и демонстрирует значительный прирост производительности оптимизированной версии.

Работа поддержана грантом российского фонда фундаментальных исследований №14-07-31090.

6. Список литературы

[1] I. Potapev, D. Kovalenko. Scene boundary localization based on contrast region analysis // Tomsk Polytechnic University, MSIT №10. 2012. – pp 60-62.

[2] O. Rosenberg OpenCL Overview AMD, 2011 // [электронный ресурс] http://www.khronos.org/assets/uploads/developers/library/overview/opencl-overview.pdf

[3] R. Farber Part 2: OpenCL – Memory Spaces // [электронный ресурс] http://www.codeproject.com/Articles/122405/Part-2-OpenCL-Memory-Spaces

[4] K. Reda A study of OpenCL image convolution optimization // [электронный ресурс] http://www.evl.uic.edu/kreda/gpu/image-convolution/

[5] H. Dong Cross-Platform OpenCL Code and Performance Portability for CPU and GPU Architectures investigated with a Climate and Weather Physics Model // Fifth International Workshop on Parallel Programming Models and Systems Software for High-End Computing, 2012. [электронный ресурс] http://www.mcs.anl.gov/events/workshops/p2s2/2012/slides/Han_Dong_p2s2_icpp_workshop.pdf