ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ CUDA ДЛЯ НЕГРАФИЧЕСКИХ ВЫЧИСЛЕНИЙ НА GPU

Красилов А. А.
Кафедра вычислительной техники, Пензенский государственный университет
Пенза, Российская Федерация
E-mail: nakeud@rambler.ru

В работе изучаются способы ускорения вычислений на персональных компьютерах, в особенности распараллеливание обработки многомерных массивов однотипной информации. Выбирается способ повышение быстродействия за счет перенесения вычислений на графический процессор (GPU). Выбирается современная технология NVIDIA CUDA, которая позволяет программисту задействовать ресурсы GPU для ускорения вычислений. Составляется программа, которая показывает возможность повышения быстродействия за счет перенесения обработки многомерных массивов однотипных данных на GPU.

Введение

В современных персональных компьютерах используются универсальные процессоры (CPU). Они предоставляют множество функций и приемлемое быстродействие. Но в сферах, где компьютер преимущественно используется для математических и других расчетов часто быстродействие можно увеличить. Сопроцессоры для обработки чисел с плавающей точкой ускоряли определенную область вычислений. С развитием компьютерной графики появилась необходимость обработки векторов и массивов однотипных данных. На помощью к CPU пришла технология MMX, которая в последствии сменилась технологиями SSE. Эти технологии использовали SIMD (одиночный поток команд, множество потоков данных), что давало прирост производительности при обработки векторов графический информации. Но прирост производительности можно было так же получить при использовании графического процессора (GPU) для такого рода вычислений, так как GPU специально предназначены для быстрой обработки графической информации. Так GPU стала неотъемлемой часть персональных компьютеров наряду с CPU, и именно GPU берет на себя обработку текстур и сложных графических объектов, представляющих собой вектора и массивы однотипных данных, и дает высокую производительность. Через некоторое время стали появляться специальные шейдерные программы, которые задействовали GPU не только для обработки изображений в 3D приложениях, но и применялись в других параллельных расчётах. В этих программах данные к видеочипу передавались в виде текстур, а расчётные программы загружались в виде шейдеров. Недостатками такого метода является сравнительно высокая сложность программирования, низкая скорость обмена данными между CPU и GPU. Но создание более совершенного GPU общего назначения (GPGPU) получило сильный толчок. В последствии, компания NVIDIA выпустила платформу CUDA (Compute Unified Device Architecture). В отличие от предыдущих моделей программирования GPU, она была выполнена с учётом прямого доступа к аппаратным возможностям видеокарт. CUDA – C-подобный язык программирования со своим компилятором и библиотеками для вычислений на GPU.

I. Постановка задачи

Определенные научные сферы и производственные отрасли используют многомерные математические расчеты и вычисления. Их многомерность позволяет распараллеливать вычисления для достижения большей производительности. Но использование CPU для такого рода параллельных расчетов ограничивает быстродействие. Если же перенести эти многомерные вычисление на GPU с изначально параллельной организацией работы, то достигается большая степень распараллеливания и, вследствие чего, большая производительность.

Цель настоящей работы – применение возможностей технологии CUDA для перенесения расчетов обработки массивов аналоговой информации, полученной с различного рода датчиков, на GPU для увеличения скорости обработки и производительности. Использование технологии CUDA накладывает определенные требования к используемому персональному компьютеру. NVIDIA CUDA есть у чипов G8x, G9x и GT2xx, применяемых в видеокартах GeForce серий 8, 9, 200 и всех последующий. CUDA доступна на 32-битных и 64-битных операционных системах Linux, Windows и MacOS X. Чтобы использовать эту технологию в своих программах, ваш компьютер должен соответствовать этим требованиям, т.е. операционная система, видеокарты фирмы NVIDIA, семейство карт, которое поддерживает данную технологию. Если видеокарта поддерживает CUDA, то желательно скачать и установить последнюю версию драйвера. Даже если поддержка отсутствует, создавать CUDA-приложения возможно, просто они будут запускаться в режиме эмуляции и нельзя будет достичь прироста производительности.

В состав CUDA входят runtime библиотеки:

- общая часть, предоставляющая встроенные векторные типы и подмножества вызовов RTL, поддерживаемые на CPU и GPU;
- CPU-компонента, для управления одним или несколькими GPU;
- GPU-компонента, предоставляющая специфические функции для GPU.

Основной процесс приложения CUDA работает на универсальном процессоре (host), он запускает несколько копий процессов kernel на видеокарте.

II. Работа программы

Составленная программа засекает время обработки, вызывает функцию расчета адаптивных порогов, получает время обработки. Функция обработки адаптивных порогов составлена с использованием технологии CUDA. В ней многомерные вычисления перенесены на видеокарту. Выделяется память на видеокарте с использованием функций CUDA, указываются измерения распараллеливания (количество блоков и нитей), вызывается ядро обработки (kernel), копии которого будут параллельно запущены на множествах ядер видеокарты с разными итераторами. Используются специальные итераторы технологии CUDA, с помощью которые на разные ядра поступает копия процесса kernel, использующая разные ячейки обрабатываемых массивов. Полученные результаты находятся в памяти видеокарты. Далее происходит копирование данные из памяти GPU в память CPU с использованиями функции CUDA. Память видеокарты освобождается, а результаты в памяти CPU передаются на выход функции. Пользователь программы на экране может видеть полученное время обработки. Была написана программа, производящая расчеты без использования GPU. Оба кода объединены в одну программу, и пользователь может видеть на экране время расчетов с технологией CUDA и без нее. Средний прирост производительности составляет 30

Модификация данного кода может дать куда лучшие показатели производительности. Функция расчетов по замыслу может вызываться множество раз, что увеличивает многомерность расчетов. Память GPU выделяется и освобождается каждый раз при вызове функции, что снижает быстродействие. Если выделить память GPU единожды, вне функции и передать внутрь, то производительность будет увеличена. При этом функция расчета адаптивных порогов будет вызываться множество раз, и каждый раз будет использовать одну и ту же память. Но если затратить больше ресурсов видеокарты, выделить память под все обрабатываемые массивы сразу, а расчет вынуть из функции и объединить с циклом вызова функции, что добавит одно измерение к обрабатываемым массивам, то можно будет распараллелить сразу весь расчет, а не его единицу. Это даст большой прирост производительности, но снизит читаемость и модульность кода.

Использование GPU имеет некоторые ограничений. Одно из них – из процесса kernel, выполняемого на ядрах видеокарты нельзя вызывать функции. Это ограничение приводит к тому, что для достижения производительности приходиться жертвовать вложенностью и модульностью кода. Но если производительность – это главный критерий, то это приемлемые жертвы.

Заключение

Рассмотрены тенденции ускорения вычислений с использованием компьютера. Выбрана технология CUDA, которая позволяет повысить быстродействие обработки многомерных векторов и массивов однотипных данных. Технология была протестирована в сравнении с обычной программой, был получен прирост производительности, выявлены некоторые ограничения технологии, которые в будущем могут быть устранены, выявлены некоторые недостатки теста, исправление которых могут дать еще больший прирост производительности.

1. NVIDIA CUDA – неграфические вычисления на графических процессорах [Электронный ресурс] – Режим доступа: http://www.ixbt.com/video3/cuda-1.shtml.
2. CUDA Toolkit Documentation [Electronic resource] – Mode of access: http://docs.nvidia.com/cuda/index.html.
3. Создание простого приложения CUDA в Visual Studio 2010 [Электронный ресурс] – Режим доступа: http://www.mezhov.com/2011/09/cuda-visual-studio-2010.html.