ПРИМЕНЕНИЕ ТЕХНОЛОГИИ CUDA ДЛЯ ОБРАБОТКИ РАДИОЛОКАЦИОННЫХ ДАННЫХ

Голутвин Р. И., Красилов А. А.
Кафедра вычислительной техники, Пензенский государственный университет
Пенза, Российская Федерация
E-mail: g-rom-i@yandex.ru

Алгоритмы обработки радиолокационной информации предъявляют высокие требования к производительности вычислительных машин. Тем не менее, эти алгоритмы имеют неотъемлемый параллелизм обработки данных, который обеспечивает большую производительность на параллельных архитектурах. В данном докладе рассматривается технология CUDA как средство для эффективной обработки радиолокационных данных.

Введение

Радиолокационный комплекс авиационной системы дальнего радиолокационного обнаружения представляет собой сложнейшую радиотехническую систему, при проведении летных испытаний которого применяются технологические аппаратно-программные средства, обеспечивающие регистрацию радиолокационных сигналов бортовых РЛС и последующий анализ зарегистрированной информации на ЭВМ в целях оценки эффективности обнаружения целей в условиях помех различного вида. Подсистема анализа радиотехнических сигналов (ПАРТС) это приложение, выполняющее трудоемкие операции по визуализации и анализу больших объемов радиолокационных данных, полученных во время летных испытаний [1]. Радиолокационная информация представляет собой множество двухмерных матриц, называемых кадрами, полученных в результате БПФ-преобразования прошедших дискретизацию эхо-сигналов. Каждый элемент такой матрицы соответствует уровню сигнала на заданной частоте и элементе дальности. Обработка радиолокационных данных по большей части заключается в применении некоторого алгоритма к каждому из значений уровня сигнала (например, применение логарифмического преобразования). Этот факт дает основание предположить, что технология CUDA, которая позволяет выполнять один и тот же алгоритм для разных данных на множестве параллельных нитей (потоков), будет эффективна для ускорения обработки радиолокационной информации.

Технология CUDA

Графический процессор (GPU) является параллельной архитектурой, которую активно используют для обработки как графической, так и неграфической информации. Первоначально графические ускорители предназначались для ускорения обработки 3D-графики. После нескольких эволюционных этапов, их архитектура позволила применять фрагментные процессоры для обработки больших массивов данных. Возникновение архитектуры, позволяющей производить параллельные вычисления данных больших объемов, привело к потребности в средствах разработки GPGPU-приложений (General-Purpose computing on Graphics Processing Units). В качестве такого средства выступает CUDA (Compute Unified Device Architecture).

Предложенная компанией Nvidia технология CUDA предназначена для разработки приложений для массивно-параллельных вычислительных устройств [2]. Ядро CUDA-программы представляет собой алгоритм, который выполняется каждой нитью в GPU-процессоре. Ядро запускается на выполнение с центрального процессора, при этом в GPU отправляются входные данные. После завершения вычислительных операций происходит копирование результатов из глобальной памяти GPU в память CPU. Такая структура позволяет проводить огромное количество однотипных вычислений одновременно, что при обработке больших объемов данных дает значительный прирост производительности по сравнению с CPU.

Обработка радиолокационной инофрмации с помощью CUDA

В ПАРТС при обработке радиотехнических сигналов применяются такие алгоритмы, как транспонирование матрицы, применение пороговой обработки, контурной обработки, вычисление цветового эквивалента для значения матрицы и др. Все эти алгоритмы объединяет то, что над каждым элементом матрицы выполняются однотипные действия. Благодаря этому, данные алгоритмы можно значительно ускорить, распределив обработку отдельных элементов на параллельные потоки. В данном докладе рассматривается распараллеливание алгоритма пороговой обработки радиолокационных данных с помощью CUDA.

Пороговая обработка заключаются в сравнении модулей сигналов с порогом P. При этом существует 2 вида порогов: ручной порог и адаптивный порог. Ручной порог – числовое значение, которое задается оператором. Если значе-

ние уровня сигнала ниже этого порога, то этот уровень сигнала не будет учитываться при обработке и визуализации радиолокационных данных. Адаптивный порог – числовое значение, которое вычисляется как сумма значений модуля сигнала на строке дальности частотного канала, поделенная на число элементов дальности в данном частотном канале, и у множенная на коэффициент порога, задаваемый оператором. В докладе описывается программа расчета адаптивных порогов, написанная с помощью технологии CUDA.

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 процентов.

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 ограничения. 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

В результате тестирования обработки радиолокационных данных с помощью технологии CUDA на примере вычисления адаптивных порогов, было выяснено, что благодаря распараллеливанию вычислений можно значительно ускорить процесс обработки радиолокационной информации. Данные выводы дают основание использовать технологию CUDA для реализации других алгоритмов обработки радиолокационных данных, требующих объемных однотипных вычислений.

1. Голутвин Р.И., Коннов Н.Н., Федюнин Р.Н. Программные средства анализа сигналов импульсно-доплеровской РЛС. / Труды IX Международной научно-технической конференции «Новые информационные технологии и системы» - Пенза : Изд-во ПГУ, 2010. – Ч. 1. – 256 с.
2. Борееков А.В., Харламов А.А. Основы работы с технологией CUDA. - М.: ДМК Пресс, 2010. – 232 с.
3. NVIDIA CUDA – неграфические вычисления на графических процессорах [Электронный ресурс] – Режим доступа: http://www.ixbt.com/video3/cuda-1.shtml.
4. CUDA Toolkit Documentation [Electronic resource] – Mode of access: http://docs.nvidia.com/cuda/index.html.
5. Создание простого приложения CUDA в Visual Studio 2010 [Электронный ресурс] – Режим доступа: http://www.mezhov.com/2011/09/cuda-visual-studio-2010.html.