ПРИМЕНЕНИЕ ТЕХНОЛОГИИ CUDA ДЛЯ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЯ.

Новинский Н. Б.

Федеральное государственное унитарное предприятие «Главный радиочастотный центр»
(ФГУП «ГРЧЦ»)

Для получения аттракторов, близких к исходным изображениям, методом полного перебора, необходим огромный вычислительный ресурс.

Поскольку вычислительные способности, существующих на данный момент вычислительных систем, например, на базе центральных процессоров от Intel явно недостаточно, было принято решение попробовать многопроцессорные решения на базе технологии CUDA от nVidia.

Введение.

Фрактальное сжатие основано на получении некоторого преобразования, применение которого несколько раз к произвольному изображению позволяет получить новое изображение, не меняющееся при очередном применении этого преобразования. Такое преобразование называется фрактальным, а полученное таким образом изображение называется аттрактором. Впервые такие преобразования описаны Мандельбротом в [1]. В [2] Барнсли подробно описал один из возможных алгоритмов получения преобразования. Он заключается в разбиении исходного изображения на множество квадратных ранговых областей и подбора для каждой похожего домена. Домены при этом уменьшаются в два раза (до размера ранговой области), поворачиваются при этом в восемь разных положений, приводятся по яркости и контрасту, и сравниваются в среднеквадратичном смысле с ранговой областью. Следует заметить, что такая методика поиска чрезвычайно затратна по вычислительным ресурсам, поскольку требует многократного полного перебора всех возможных доменов. Причем все существующие методики ускоренных расчетов, как правило, дают худший результат.

В проводившихся ранее исследованиях на тему фрактального сжатия, использовались изображения небольших размеров (в основном 256x256, 512x512). Не в последнюю очередь это было связано с ограниченными вычислительными ресурсами. Однако, в настоящее время, в связи с широким внедрением в массовое пользование фотографической техники с большими матрицами и повсеместным внедрением новых стандартов представления видеоинформации с большим разрешением, обработка больших изображений приобретает особенную актуальность. С другой стороны чем больше изображение, тем выше возможность подобрать действительно похожую пару домен – ранговая область, что должно благотворно сказываться на качестве полученного аттрактора.

Работать предполагалось с изображениями нескольких размеров. В таблице 1 в третьей колонке представлены размеры наиболее известных видеоформатов. В первой колонке приводится сленговое название (оно коррелирует с числом строк), к которому будут отсылки в тексте данной статьи. В последнем столбце размеры изображений на которых планировалось производить расчеты. Размеры выбраны исходя из кратности оных размерам ранговых областей. Формат 0,5К хоть и не совпадает ни с одним видеоформатом, однако близок по размерам к изображениям, использовавшимся для экспериментов с изображениями ранее, и именно на них и проводилась тестовая отработка алгоритмов.

0,5K			960x544
1K	HDTV	1920x1080	1920x1088
2K	Quard HDTV	3840x2160	3840x2176
3K		5760x3240	5760x3264
4K	UHDTV	7680x4320	7680x4352

Таб.1 Наиболее известные форматы видеоизображений.

Следует заметить, что в данной работе вообще говорить о сжатии не приходится. Поскольку результатом расчетов является текстовый файл (для удобства анализа), из которого можно было получить аттрактор максимально близкий к исходному изображению.

	Размеры ранговых областей		
	4x4	8x8	16x16
CPU	83 902	62 353	54 996
GPU, блоки в одну нить, использование константной памяти	11 449	8 935	7 964
GPU, блоки со множественными нитями, использование разделяемой памяти	2 518	678	393
GPU, оптимизация вывода	428	180	393

Таб.2. Время получения аттрактора в сек. для изображения 0.5К.

Расчеты на CPU.

Алгоритм подбора домена для конкретной ранговой области подробно описан в [2]. Это алгоритм - один из наиболее широко представленных в литературе и исследованиях. Когда он был реализован “в лоб” на одном ядре на процессоре Intel Core Quard CPU Q9300 2,5 ГГц под ОС Linux , стало ясно, что это даже для современного процессора слишком большая задача. Были произведены работы по ускорению расчетов. Во-первых, вместо получения каждого отдельного домена, усреднением четырех соседних пикселов в один и поворотом домена в нужное положение, были рассчитаны 32 плоскости, из которых брались исходные значения для сравнения с ранговой областью. Это позволило сократить время расчетов примерно на 30 процентов. В табл. 2, в первой строке представлены времена расчетов для этого варианта алгоритма для изображения 0.5К для ранговых областей разного размера. Для справки, в сутках 86400 сек. Остальные попытки ускорить расчеты, например предварительным расчетом промежуточных сумм особого успеха не принесли, поскольку давали не более10-15 процентов уменьшения времени расчетов, но массивы, которые получались, категорически не влезали в оперативную память. Конечно, возможно было разделить задачу на все 4 ядра, и получить ускорение почти в 4 раза, однако, это тоже проблемы не решало, поскольку при росте линейного размера изображения в два раза время расчетов вырастает в 16 раз и изображение 4К должно обсчитываться примерно 1000 суток даже на 4-х ядрах.

На этом этапе было принято решение попытаться реализовать алгоритм на графическом сопроцессоре. Собственно реализации данного алгоритма на графическом процессоре посвящена данная статья. В качестве тестового варианта была выбрана видеокарта от nVidia на чипсете GeForce 250, как недорогое и достаточно производительное решение.

Расчеты с применением технологии CUDA.

Несколько слов о вычислениях на технологии CUDA.

Процессоры объединены в мультипроцессоры по 8 штук. Одна вычислительная задача в терминах CUDA называется нить. Они могут выполняться параллельно. Нити объединяются в блоки. Число нитей в блоке может быть, в достаточной степени, произвольным.

Плата обладает набором памяти различного назначения и свойств и грамотное использовании всего этого многообразия несомненно обеспечит успех. Память бывает глобальной, константной, разделяемой. Глобальная память большая (несколько гигабайт), не кешируемая и чтение одного числа занимает 400-600 тактов процессора. Что безусловно тормозит процесс расчетов. Зато все нити всех блоков имеют к ней доступ. Ускорить чтение можно следующим образом. Если каждая нить читает последовательно числа из глобальной памяти, причем все они принадлежат одному блоку, этих нитей 16 и чтение осуществляется с начала банка (по 16 чисел), то тогда скорость чтения увеличивается в 16 раз.

Константная память тоже как и глобальная обеспечивает доступ с любой нити любого блока, однако она кешируемая и маленькая. Там возможно держать небольшие объемы данных (десятки килобайт), которые нужны во всей программе. Чтение оттуда происходит достаточно быстро.

Разделяемая память маленькая (десятки килобайт), и доступ к ней возможен только из нитей одного блока. Зато очень быстрая.

В общем случае программа на CUDA состоит из пяти частей.

1. Инициализация программы на CPU.
2. Загрузка массивов данных на видеоплату за счет CPU.
3. Выполнение расчетов на GPU (ядро в терминах CUDA).
4. Выгрузка массивов данных в основную память за счет CPU.
5. Окончательная обработка данных и вывод результатов на CPU.

В данном конкретном случае пункты 2-4 выполняются неоднократно.

В начале задача была написана следующим образом. Бралась очередная ранговая область, и помещалась в константную память (пункт 2). В глобальную память подгружалась плоскость(пункт 2). В каждом блоке была только одна нить, которая считала соответствие ранговой области к очередному домену(пункт 3). Результатом расчета по каждой паре домен – ранговая область являлись 3 числа типа float, это коэффициенты преобразования для яркости и контраста и среднеквадратичная разница домен – ранговая область. Они выгружались в основную память (пункт 4). Затем из всех этих значений среднеквадратичной разницы для данной ранговой области выбирается минимальное (пункт 5, частично). Потом перебираются все ранговые области, а потом перебираются все плоскости. Результаты этих расчетов представлены в табл. 2 на второй строке. Как видно это уже замечательный результат. Ускорение почти на порядок.

На следующем этапе была введена множественность нитей в рамках одного блока, а количество блоков соответственно уменьшилось. Количество нитей в блоке было выбрано как 16 штук по горизонтали и одна высота ранговой области по вертикали. Каждая нить из блока копирует из глобальной памяти в разделяемую четыре значения в результате получается матрица, которая копирует участок плоскости размером 32 точки по горизонтали и два размера ранговой области по вертикали. Расчет производится уже на основе матрицы, которая лежит в разделяемой памяти. Результат представлен в строке 3 в табл.2. Результат расчетов для ранговых областей 16x16 получился еще луче. Время расчета для ранговых областей 4x4 оставалось все таки слишком велико.

Были проведены замеры затрат времени на различные операции. Выяснилось, что на выполнение самого ядра уходит примерно пятая часть времени. Еще три пятых времени уходит на копирование массивов с видеоплаты в основную память. И еще пятая часть времени уходит

на перебор массива в основной памяти на CPU, с целью обнаружения минимального значения для среднеквадратичной разницы. Остальные операции во всей программе занимали по времени не более одного процента.

По результатам этого исследования была произведена модернизация программы. Сортировка результатов внутри блока была вынесена на GPU. За счет этого были уменьшены выходные массивы в несколько десятков раз (от 64 до 256 раз в зависимости от размеров ранговой области). Это благотворно сказалось на результатах (нижняя строчка в табл.2).

Выводы.

1. Задача фрактального сжатия, благодатна в плане обработки на GPU поскольку хорошо распараллеливается и имеет значительные вычислительные затраты при относительно небольших входных и еще меньших выходных массивах.

2. Применение GPU позволяет достаточно резко (200-300 раз!) ускорить решение подобного рода задач. Расчетное время для обработки изображения 4К должно составить порядка 20 суток, что вполне приемлемо.

3. На данный момент существуют видеокарты от nVidia на новом 580-ом чипсете. Ожидается, что их применение и может дать выигрыш минимум в 2-3 раза. Помимо этого есть профессиональные решения под маркой TESLA, которые должны обладать еще большей производительностью. Возможно также дополнительное распараллеливание за счет применения нескольких видеокарт в рамках одной вычислительной системы.

Литература

1. Mandelbrot B.B. The Fractal Geometry of Nature / W.H.Freeman, NY, 1983
2. Barnsley M.F. Fractal Image Compression, SIGRAPH’92 Course Notes, 1992
3. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA.- М.: ДМК Пресс, 2010.
4. CUDA Reference Manual 2.3. July 2009

USING OF CUDA TECHNOLOGY FOR FRACTAL IMAGE COMPRESSION.

Abstract The biggest problem of fractal compression is tremendous time for encoding of image. In that work we used CUDA technology for brute force attractor searching and does it with considerable success.