С.А. Бибиков, А.В. Никоноров, В.А. Фурсов, П.Ю. Якимов

Самарский государственный аэрокосмический университет
им. С.П. Королева

РЕКУРРЕНТНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ В РАСПРЕДЕЛЕННОЙ МАССИВНО-МНОГОПОТОЧНОЙ CUDA-СРЕДЕ[1]

Постановка задачи. В последние годы наряду с совершенствованием и ростом производительности мощных суперкомпьютеров наметилась устойчивая тенденция развития технологий параллельных вычислений на персональных компьютерах. Эта тенденция обусловлена существенным технологическим прорывом по крайней мере в трех направлениях.

Во-первых, это появление настольных CPU с возможностью параллельного выполнения программ. В настоящее время для персональных компьютеров доступны процессоры *i*7 с параллельным исполнением 8 потоков. Вторым технологическим прорывом является представленное в 2007 году компанией Nvidia альтернативное направление развития массивно-параллельных вычислений для настольных компьютеров с использованием GPU. Третий фактор, позволяющий всерьез говорить о настольных суперкомпьютерных системах, – это повышение скорости коммуникаций по системной шине для платформы x86 с развитием стандарта PCI Express. Скорость коммуникаций между узлами всегда была узким местом кластерных систем, снижающим их эффективность. Принятие подавляющим большинством производителей стандарта *Infiniband* позволяет строить весьма недорогие вычислительные системы со скоростью обмена данными между узлами до 5 GB/s.

Совокупность указанных факторов стала мотивом возрастающего интереса к созданию компактных, недорогих и вместе с тем высокопроизводительных систем обработки и анализа

[1] Работа выполнена при финансовой поддержке РФФИ (проект № 09-07-00269-а) и программы Президиума РАН.

изображений нового поколения для систем видеоконтроля, видеонаблюдения, подготовки цифровых изображений к печати и др. В работах [1–3] рассматривались примеры создания компактных распределенных систем обработки изображений. В частности, в работе [1] описана реализация алгоритма коррекции цветных цифровых изображений в локальной сети компьютеров посредством специально разработанной GRID-системы, реализованной на Java с native-модулями. В этой работе показана возможность существенного ускорения одного из наиболее затратных по времени процессов подготовки цветных изображений к печати. В работах [4–5] приведены примеры дальнейшего ускорения предпечатной подготовки изображений за счет реализации алгоритмов обработки изображений в CUDA-среде.

Наиболее популярным и широко используемым во многих технологиях обработки изображений является метод скользящего окна. Известны быстрые рекуррентные (например, параллельно-рекурсивные) алгоритмы обработки изображений скользящим окном, реализованные на традиционных аппаратных средствах [6]. Создание таких же эффективных процедур на основе CUDA-технологий, реализуемых с использованием графических ускорителей NVIDIA, к сожалению, пока еще остается проблемной задачей. Связано это с тем, что вычислительная среда CUDA имеет более сложную с точки зрения программирования структуру по сравнению с многопоточной CPU-средой или параллельной средой MPI. Основные структурные особенности среды CUDA следующие.

Во-первых, CUDA-среда имеет двухуровневую многомерную топологию исполнения потоков, которая может описывать десятки тысяч потоков. По аналогии с термином *массивно-параллельная* среда такую среду можно назвать *массивно-многопоточной*.

Во-вторых, потоки могут обмениваться данными только посредством сравнительно небольшого объема общей памяти. Такой обмен возможен в пределах небольшой области локальной связности – блока, включающего (для выпускаемых в настоящее время GPU) не более 512 потоков. Кроме того, когерентность данных в общей памяти достигается только путем

приостановки всех потоков, т.е. упорядоченный обмен данными невозможен. Указанные ограничения создают серьезные проблемы при реализации в среде CUDA рекурсивных алгоритмов (сжатия данных, фильтрации и др.) [7].

Наконец, еще одной важной особенностью является то, что в CUDA-среде можно работать с несколькими видами памяти, имеющими разную область видимости и скорость доступа. Крайне важным при доступе к глобальной памяти является согласование (coalescing) адреса памяти с номером активного потока. Если условие согласования выполняется, то GPU может объединить несколько запросов в одну транзакцию [14]. Несоблюдение этого условия, может привести к падению производительности почти в 8 раз [8, 9].

Многие базовые алгоритмы обработки данных требуют существенной переработки с учетом указанных особенностей. Например, в работе [10] рассмотрен адаптированный для CUDA-среды алгоритм префиксного суммирования. В настоящей статье, являющейся развитием работ [1–5], из всего комплекса рассмотренных в них задач выделены наиболее характерные и сложные для реализации на GPU-процессорах задачи: обработка изображений скользящим окном и организация передачи видеоданных в реальном времени. В частности, рассмотрены основные особенности построения указанных алгоритмов и приводится краткое описание созданного в массивно-многопоточной CUDA-среде программного комплекса.

Рекуррентная CUDA-реализация обработки изображений скользящим окном. Продемонстрируем особенности реализации рекуррентных вычислительных алгоритмов в CUDA-среде на примере простейшего локального оконного фильтра, выполняющего расчет среднего значения. Простота этого алгоритма позволяет наглядно продемонстрировать все основные особенности среды CUDA. Вместе с тем иллюстрируемый на этом простом примере подход может быть легко применен к более сложным локальным оконным фильтрам, применяемым в различных технологиях обработки изображений.

Пусть алгоритм обработки скользящим окном состоит в вычислении для каждой точки изображения суммы значений отсчетов в некоторой окрестности (окне):

$$s_{x,y} = \sum_{i,j=1}^{m} L_{x+i,y+j}, \quad 0 < x, y \le N, \tag{1}$$

где N – размер изображения, m – размер окна (для простоты полагаем, что изображение и окно квадратные). При этом потребуется

$$O_1 = N^2 m^2 \tag{2}$$

арифметических операций.

При рекуррентной реализации этих вычислений в CUDA-среде существенное значение имеет согласование адреса памяти с номером активного потока. Будем считать обращение потока к глобальной памяти согласованным, если оно происходит к словам памяти размером 32 бита с адресами

$$I(t,k) = 32n \cdot k + t, \quad k = 0,1,2..., \quad t = 0,1,2,..., \tag{3}$$

где $I(k,t)$ – адрес ячейки глобальной памяти, к которой производится обращение, n – натуральное число, характеризующее особенности конкретной задачи, связанные с количеством потоков в блоке; t – номер потока внутри блока, k – номер шага некоторой итерации алгоритма, одинаковый для всех потоков внутри блока. Приведенные в спецификации CUDA [8] требования согласованности несколько более сложные и к тому же различаются для различных реализаций стандарта CUDA. Тем не менее легко проверить, что выполнение требования (3) для всех потоков внутри блока позволяют получить согласованный доступ к памяти в смысле спецификации [8].

Рекуррентный алгоритм, удовлетворяющий условию согласованности (3), может быть построен, например, как двухпроходный. При этом сначала выполняется вычисление частичных сумм по столбцам:

$$s^1_{x,y} = s^1_{x,y-1} - L_{x,y-m} + L_{x,y}, \quad s_{x,1} = \sum_{j=1}^{m} L_{x,y+j}, \tag{4}$$

а окончательная сумма получается суммированием по строкам:

$$s_{x,y} = s_{x-1,y} - s^1_{x-m,y} + s^1_{x,y}, \quad s_{1,y} = \sum_{j=1}^{m} s^1_{x+j,y}. \qquad (5)$$

Вычисления по такой схеме требуют

$$O_1 = (2N - m)^2 \qquad (6)$$

арифметических операций.

Ниже приводятся результаты сравнения эффективности предложенных реализаций. Эксперименты проводились на тестовом изображении размером 1024×1024 для окон различных размеров. В первом случае декомпозиция проводилась на 2048 блоков по 512 потоков, часть которых запускалась не одновременно. Во втором и третьем случае запускалось одинаковое количество потоков – 1024, при этом они группировались в 32 блока по 32 потока в каждом. Тестирование проводилось для CPU PIV 3Ghz и GPU Nvidia GF 9500. Результаты представлены в таблице.

Время выполнения различных алгоритмов для разных размеров окна (*m*, мс)

№ п/п		*m* = 5	*m* = 9	*m* = 15
1	CPU, не рекуррентный	283	922	2 531
2	CPU, рекуррентный	142	297	733
3	GPU, не рекуррентный	78	262	473
4	GPU, рекуррентный	23	25	27
5	GPU, рекуррентный с оптимизацией обращений к памяти	12	12	12

Как видно из таблицы, результаты для рекуррентной CUDA-реализации практически не зависят от *m*, а зависят только от стратегии использования памяти.

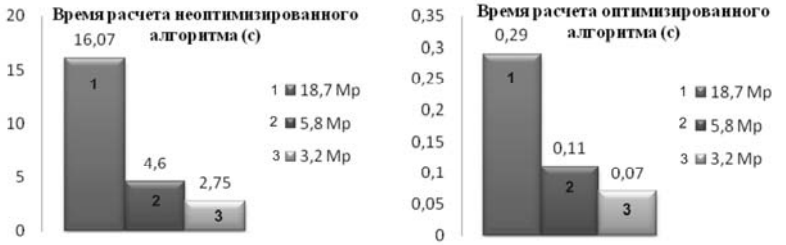
Реализация программного комплекса и результаты экспериментов. Описанная выше схема взаимодействия пото-

ков с памятью при реализации рекуррентного алгоритма использовалась при создании последних версий программного комплекса локализации и коррекции бликов на цифровых изображениях репродукций произведений живописи. Первые версии этого программного комплекса описаны в работах [2–5]. Программный комплекс имеет широкий набор функций для работы с цифровыми изображениями. Приложение работает в режиме *stand alone*, т.е. не требует установки дополнительного программного обеспечения. Благодаря использованию кроссплатформенных библиотек wxWidgets комплекс работает как на платформе Windows, так и на Unix-подобных операционных системах.

Рис. 1. Вид GUI

На рис. 1 представлен вид GUI, разработанного приложения в режиме автоматизированного поиска точечных артефактов (красные точки указывают найденные блики). В последней версии параметры поиска могут задаваться как вручную, так и автоматически путем перехода в режим адаптивного определения параметров. Кроме того, имеется возможность выполнять масштабирование изображения, выбор области поиска, а также модифицировать маску с отмеченными артефактами, сформированную алгоритмом, вручную.

В работах [4] и [5] авторами была показана возможность существенного ускорения алгоритмов обработки изображений в программной системе локализации и устранения бликов за счет использования универсальных графических процессоров NVIDIA и технологии CUDA. В частности, на графических ускорителях NVIDIA GeForce 8400 и 9500 с 8 мультипроцессорами, было продемонстрировано более чем десятикратное ускорение CUDA-реализации по сравнению с реализацией на CPU. В настоящей работе мы демонстрируем возможность дальнейшего повышения ускорения за счет использования в программной системе оптимизированных рекуррентных алгоритмов.



а *б*

Рис. 2. Время работы CUDA-алгоритма: *а* – алгоритм, опубликованный в работе [4]; *б* – результаты оптимизированного рекуррентного алгоритма

Для экспериментальной проверки скорости использовались три изображения размером 3,2, 5,8 и 18,7 мегапикселей. Для расчетов использовалась видеокарта Nvidia GeForce 9500. На рис. 2, *а* приведено время вычислений для алгоритма из работы [4], на рис. 5, *б* приведено время вычислений для алгоритма поиска бликов на изображении, использующего оптимизированную рекуррентную процедуру обработки скользящим окном. Из приведенных диаграмм видно, что в среднем, по трем экспериментам, достигнуто ускорение более чем в 45 раз.

Подчеркнем, что полученное в эксперименте ускорение получено исключительно за счет оптимизации CUDA-алгоритма.

Достигнутые показатели скорости обработки с использованием оптимизированной версии алгоритма поиска бликов открывают возможности для их более широкого применения. В частности, наряду с использованием этого алгоритма в задачах предпечатной обработки изображений, как это было сделано в [4], по-видимому, удастся использовать эту технологию также для обработки видео данных в реальном времени.

Список литературы

1. Никоноров А.В., Фурсов В.А. Предоставление сервиса управления цветовоспроизведением цветных изображений в сети интернет // Телематика–2007: тр. XIV Всерос. науч.-метод. конф. – С.-Петербург, 18–21 июня, 2007. – СПб.,2007. – С. 412–413.

2. Никоноров А.В., Фурсов В.А. Распределенная вычислительная среда коррекции цветных изображений // Телематика–2008: тр. XV Всерос. науч.-метод. конф. – С.-Петербург, 23–26 июня, 2008. – СПб., 2008. – С. 88–89.

3. Фурсов В.А., Никоноров А.В. Распределенный алгоритм коррекции точечных артефактов на цветных изображениях // Параллельные вычисления и задачи управления (PACO–2008): тр. VI Междунар. конф. Москва, 27–29 октября, 2008. – М.,2008.

4. Исследование эффективности технологии CUDA в задаче распределенной предпечатной подготовки цифровых изображений / С.А. Бибиков [и др.] // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: тр. всерос. суперкомпьютерной конф. 21–26 сентября 2009 г., г. Новороссийск. – М.: Изд-во МГУ, 2009. – С. 204–207.

5. CUDA-технология цветовой коррекции теневых искажений на цифровых фотокопиях произведений живописи / С.А. Бибиков [и др.] // Параллельные вычислительные технологии–2010: сб. тр. междунар. науч. конф. Уфим. гос. авиац. техн. ун-та (УГАТУ), г.Уфа, 29 марта – 2 апреля 2010 г. – Уфа,2010. – 656 с.

6. Методы компьютерной обработки изображений / Сойфер В.А. [и др.]. – М.: Физматлит, 2003. – 784 с.

7. Nickolls J., Buck I., Skadron K., Scalable Parallel Programming with CUDA // ACM Queue, Vol. 6. No. 2. March/April 2008. – P. 40–53. – URL: http://mags.acm.org/queue/20080304/?u1=texterity.

8. NVIDIA CUDA Programming Guide, 2010. 130 p. – URL: http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf.

9. NVIDIA CUDA Best Practices Guide, 2009. – 68 p.

10. Harris M., Parallel Prefix Sum (Scan) with CUDA, 2008. – 21p. – URL: http://developer.download.nvidia.com/compute/cuda/sdk/website/projects/scan/doc/scan.pdf.

11. Мурызин С.А., Сергеев В.В., Фролова Л.Г. Исследование эффективности двумерных параллельно-рекурсивных КИХ-фильтров // Компьютерная оптика. – М.: МЦНТИ, 1992. – Вып. 12. – С. 65–71.

12. Smart, J. Cross-platform GUI Programming with wxWidgets. – Prentice Hall, 2000. – 714 p.

13. Munshi A. Kronos OpenCL Working Group, The OpenCL Specification. – 2009. – 314 p.

14. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.

М.Г. Бояршинов, Д.С. Балабанов

Пермский государственный технический университет

ВЫЧИСЛИТЕЛЬНОЕ МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ СЖИМАЕМОЙ СРЕДЫ С ИСПОЛЬЗОВАНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА

Методы вычислительного моделирования движения газовых потоков с использованием высокопроизводительных вычислительных комплексов в настоящее время являются основным инструментом решения краевых задач механики сплошных сред, включающих системы дифференциальных уравнений