Эффективное сжатие цифровых изображений с применением высокопроизводительных вычислительных систем

С.С. Кулбаев, И.В. Бойченко, В.В. Голенков
Томский государственный университет систем управления и радиоэлектроники

При сжатии качество и размер цифровых изображений всегда являются противоречивыми критериями. В данной работе предлагается способ увеличить коэффициент сжатия изображений и при этом уменьшить потери качества. Существенное увеличение коэффициента сжатия цифровых изображений достигается путём применения параллельного алгоритма сжатия на основе фракталов с применением высокопроизводительных вычислительных систем (ВПВС).

Введение

В последние годы для достижения высокой степени сжатия изображений было предложено множество методов. Среди них очень перспективным методом сжатия является метод фрактального сжатия изображений [1]. Большую ценность представляют алгоритмы фрактального сжатия, способные работать в режиме реального времени, однако большинство из них требует много времени для обработки данных в силу трудоемкости. Более быстрые алгоритмы сокращают время работы за счёт ухудшения качества обработки данных и/или коэффициента сжатия (соотношение размер/качество). Таким образом, разработка методов и алгоритмов на основе фрактального сжатия, способных обрабатывать данные за короткий промежуток времени с высоким качеством, является актуальной задачей и представляет научный интерес. Фрактальное сжатие поддается параллельной обработке, что позволяет создавать программное обеспечение для эффективного решения данной задачи на множестве вычислительных узлов.

Организация параллельных вычислений

Параллелизм (одновременное выполнение нескольких операций обработки данных) осуществляется, в основном, введением избыточности функциональных устройств (многопроцессорности) [2]. В этом случае можно достичь ускорения процесса решения вычислительной задачи, если осуществить разделение применяемого алго-

ритма на информационно независимые части и организовать выполнение каждой части вычислений на разных процессорах. Подобный подход позволяет выполнять необходимые вычисления с меньшими затратами времени. Однако максимально возможное ускорение ограничивается числом имеющихся процессоров и количеством «независимых» частей в выполняемых вычислениях, т.е. количеством заданных рангов. Алгоритм фрактального кодирования заключается в разбиении изображения на ранговые блоки и поиске для них на этом же изображении доменных блоков более крупного размера.

Ранговые блоки покрывают все изображение и не накладываются друг на друга, доменные блоки могут накладываться друг на друга и не покрывать все изображение. Доменный блок должен максимально соответствовать ранговому блоку и обязательно быть больше него. Кодирование изображения представляет собой совокупность ранговых блоков и преобразование их в доменные [3].

В параллельном алгоритме сжатия изображений на основе фракталов изображение разбивается на несколько ранговых блоков. Каждый блок обрабатывается отдельным потоком. Обработка рангового блока подразумевает поиск соответствия данного рангового блока доменному блоку. В случае, если соответствующий доменный блок не найден, ранговый блок разбивается на несколько меньших блоков, каждый из которых, в свою очередь, обрабатывается отдельным потоком.

Многопоточная обработка (multithread processing)

Одним из преимуществ многопоточной обработки является возможность в многопроцессорных системах одновременно выполнять несколько потоков на разных процессорах, увеличивая тем самым производительность.

Многопоточная обработка изображений с помощью фрактального алгоритма происходит следующим образом. Исходное изображение разбивается на ранговые блоки и на группы доменных блоков. Затем создаются потоки, равные количеству ранговых блоков, и для каждого из ранговых блоков создаётся по потоку. Потоки работают независимо друг от друга. Каждый поток, в свою очередь, порождает по четыре потока. Порождённые потоки могут также порождать ещё по четыре потока (по мере необходимости). И так далее, до тех пор, пока не будет достигнут минимальный размер блока (рис. 1). Причём чем меньше размер блока, тем меньше потери качества изображения. При его достижении поток записывает в файл не сами данные рангового блока, а структуру данного рангового блока. В состав структуры рангового

блока входят: размер рангового блока; координаты по оси x, y; форма аффинного преобразования; три цветовые составляющие (YUV).

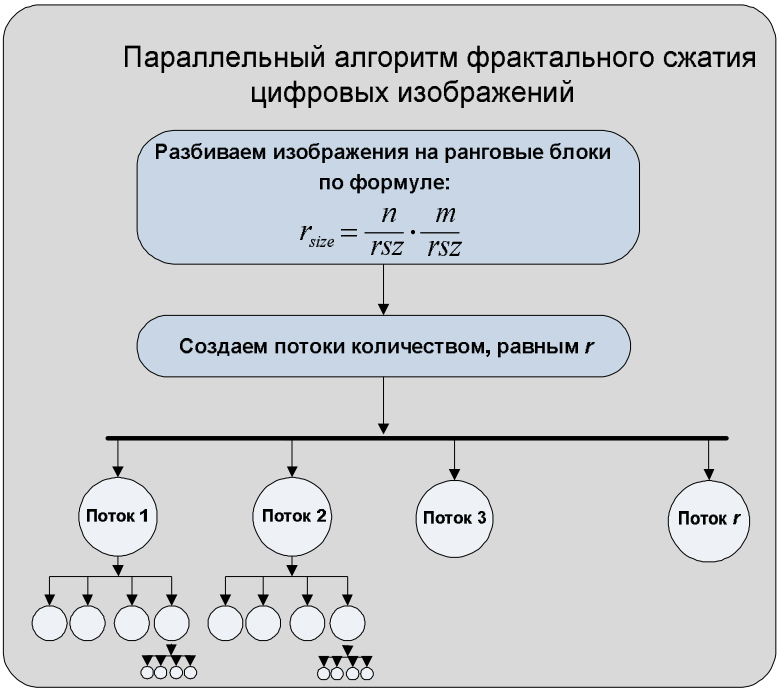


Рис. 1. Схема работы многопоточного параллельного алгоритма фрактального сжатия изображения

Результаты работы

Результаты работы приведены в таблице. В ходе исследования было использовано изображение размером 512x512 пикселей, 24 бит на пиксель. Размер изображения составляет 512x512x24 = 6 291 456 бит, или 786,4 Кбайт. В состав параметров фрактального алгоритма входят: размер рангового блока (8), шаг поиска домена (4), погрешность (среднеквадратичное отклонение 0,01), режим поиска наилучшего домена в пределах среднеквадратичного отклонения. В качестве вычислительных систем был использован четырехъядерный процессор Intel Core i7, поддерживающий до 8 вычислительных потоков на четырех физических ядрах 2,8 ГГц.

Результаты работы фрактального алгоритма и JPEG на четырехъядерном процессоре

Формат	Размер тестового изображения	PNM PSNR, дБ	Время сжатия	Размер выходного файла
BMP	786,4 Кбайт (512x512)	-	-	786,4 Кбайт
JPEG	786,4 Кбайт (512x512)	53.49	0.44 с (ImageMagik один поток)	63 Кбайт
Параллельный алгоритм фрактального сжатия	786,4 Кбайт (512x512)	42.73	0.24 с (многопоточное)	46,3 Кбайт

Также была проведена серия экспериментов на увеличение производительности вычислительных систем в зависимости от количества используемых ядер процессора (рис. 2). Частота одного ядра 2,8 ГГц. Как видно из графика (рис. 3), рост ускорения по мере увеличения числа ядер замедляется.

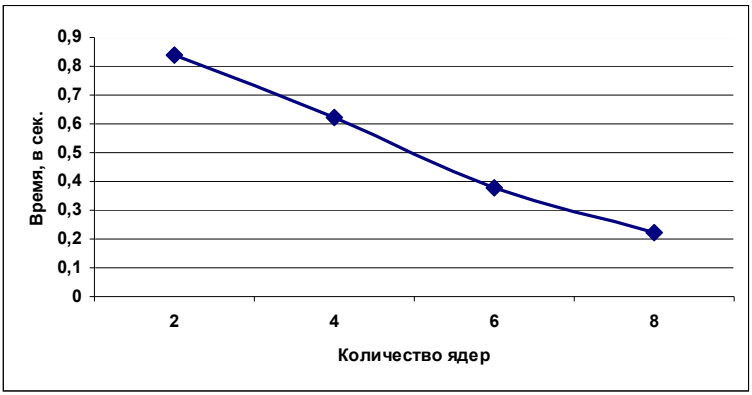


Рис. 2. Время выполнения на многоядерных вычислительных системах

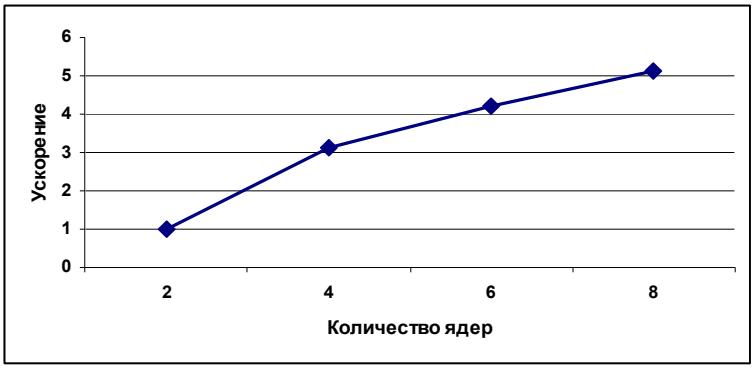


Рис. 3. Коэффициент ускорения от числа ядер

Оптимизация с помощью развёртки циклов

Одним из методов оптимизации программ является метод развертки циклов, который позволяет за счет увеличения количества исполняемого кода увеличить скорость выполнения программ. Техника разворачивания циклов в общем случае сводится к уменьшению количества итераций за счет дублирования тела цикла соответствующее число раз.

При параллельной обработке изображений на основе фракталов одним из основных элементов является итератор. В процессе итерации указывается опорное множество отсчетов изображения, по которым осуществляется итерация, а также задаются тип и параметры преобразования (т.е. угол поворота, зеркальное отражение и т.д.), выполняемого на каждой итерации, и возможные связи с другими параллельно выполняемыми итераторами. То есть параллельно вычисляются яркость, оттенок и контрастность доменного блока, сопоставляемого с ранговым блоком. Процесс, описывающий типовую операцию обработки изображения, представляет собой параллельную композицию итераторов различного типа.

Компилятор gcc (GNU Compiler Collection) поддерживает несколько уровней оптимизации (O1 – минимальная оптимизация, O2 – умеренная оптимизация, O3 – агрессивная оптимизация). Компилятор gcc разворачивает циклы на уровне оптимизации – O2 с ключом **-funroll-loops**, поддерживающим только цикл типа for. Ключ **-funroll-all-loops** поддерживает все виды циклов (например, do/while) [4, 5].

Запуск программы производился в различных вариантах – с установленным флагом оптимизации при компиляции и без него. При

сравнении результатов запуска вариантов программы с установленным флагом оптимизации при компиляции и без включения данного флага можно сделать следующий вывод: программа, скомпилированная с флагом – O2, выполняется в среднем в 1,5 раза дольше (24 мс), нежели программа, скомпилированная с этим же флагом и с развёртыванием циклов (18 мс).

Заключение

В работе реализован многопоточный параллельный алгоритм сжатия цифровых изображений на основе фракталов с применением ВПВС. Так как каждый ранговый блок изображения в алгоритме обрабатывается независимо друг от друга, то можно достичь коэффициента ускорения, близкого к числу ранговых блоков (разумеется, если число вычислительных процессов приблизительно равно числу ранговых блоков, учитывая погрешность среднеквадратического отклонения). Если число процессов превышает количество ранговых блоков, то коэффициент ускорения не будет расти. При проведении оптимизации необходимо в первую очередь учитывать возможности компилятора и аппаратуры (тип процессора и его архитектуру) для получения наилучших показателей производительности.

Литература

1. **Ватолин Д.С.** Фрактальное сжатие изображений // ComputerWorld-Россия. – 1996. – № 6 (23) .
2. **Гергель В.П.** Основы параллельных вычислений для многопроцессорных вычислительных систем / В.П. Гергель, Р.Г. Стронгин. – Н. Новгород: ННГУ, 2003. – 2 изд.
3. **Уэлстид С.Т.** Фракталы и вейвлеты для сжатия изображений в действии: Учеб. пособие / С.Т. Уэлстид. – М.: Триумф, 2003. – 320 с.
4. **Kahle J.A., Day M.N. et al.** Introduction to the Cell multiprocessor // IBM Journal of Research and Development. – 2005. – Vol. 49, № 4/5.
5. Software Development Kit for Multicore Acceleration. Version 3.0. Programming Tutorial. DRAFT // IBM. – 2007.