

# СИСТЕМА ПАРАЛЛЕЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ EPSILON

А. В. Симаков<sup>1,\*</sup>

<sup>1</sup> Сыктывкарский государственный университет

В данной статье описывается разработанная автором система параллельного сжатия изображений EPSILON. Система базируется на вейвлетных преобразованиях, а так же промышленных стандартах для распараллеливания сложных вычислений. Программа апробирована на различных вычислительных комплексах и продемонстрировала хорошую производительность и масштабируемость. Полные исходные тексты программы EPSILON доступны для загрузки через Интернет<sup>1</sup>.

## 1. ВВЕДЕНИЕ

В данной статье описывается разработанная автором система параллельного сжатия изображений EPSILON. Система базируется вейвлетных преобразованиях [1, 2], а так же промышленных стандартах для распараллеливания сложных вычислений [3].

Благодаря применению вейвлетных методов программа EPSILON имеет отличное соотношение размер/качество и позволяет точно контролировать bit-rate изображения. Другими словами, пользователь может заранее с высокой точностью указать желаемую степень сжатия.

При кодировании данные упорядочиваются особым образом: вначале передается наиболее важная для человеческого восприятия информация, а затем уже малозаметные контуры и детали. Такой способ упорядочивания называется *прогрессивным*. При прогрессивной передаче, декодер, получая данные, может постепенно улучшать и уточнять изображение. Это гораздо предпочтительнее чем наблюдать как изображение последовательно прорисовывается сверху–вниз.

---

<sup>1</sup> <http://epsilon-project.sourceforge.net>

\* Electronic address: [xander@syktsu.ru](mailto:xander@syktsu.ru)

Кроме уже перечисленных, программа EPSILON обладает следующими характеристиками:

- Поддержка более 30 различных вейвлетных фильтров
- Поддержка огромных изображений (десятки гигабайт)
- Поддержка потоков стандарта POSIX
- Поддержка стандарта Message Passing Interface
- Устойчивый к повреждениям формат хранения
- Открытый исходный код (GNU GPL)

## 2. ПРИНЦИП РАБОТЫ

В этом разделе изложены основные принципы работы программы. Поскольку алгоритмы кодирования и декодирования изображения практически симметричны, приведем лишь схему кодирования.

**Исходное изображение.** На вход подается полноцветное или полутоновое изображение произвольного размера. На выходе мы хотим получить сжатое представление изображения, имеющее заранее заданный размер и обладающее заданными характеристиками.

**Разбиение изображения на блоки.** Поскольку изображение может оказаться очень большим, обрабатывать его необходимо поблочно. Размер и форма блока в значительной степени влияют на производительность и эффективность работы всей системы. Программа использует квадратные блоки размера  $2^N$  либо  $2^N + 1$ . По умолчанию  $N = 8$ .

**Уменьшение граничных искажений.** Обработка изображения блоками очень удобна, однако имеет один недостаток: при очень высоких степенях сжатия становятся заметными искажения на стыках блоков. В своей работе [4] авторы предлагают очень простой и эффективный способ для уменьшения граничных

искажений: брать блоки размер которых не четен и первый коэффициент разложения — низкочастотный. Поскольку низкочастотные (НЧ) и высокочастотные (ВЧ) коэффициенты чередуются, а общее их количество нечетно, то и последний коэффициент разложения также будет НЧ. Отметим, что если размер блока взять равным  $2^N + 1$  то требуемые условия будут выполнены на любом уровне вейвлетного разложения.

**Преобразование цветовых пространств.** Блоки изображения переводятся из цветового пространства  $RGB$  в пространство  $YCbCr$ . После преобразования все три компоненты сжимаются независимо друг от друга. Причем хроматические компоненты  $Cb$  и  $Cr$  можно сжимать с гораздо большими потерями, чем компоненту  $Y$ . Так, на кодирование компоненты  $Y$  по умолчанию выделяется 90% общего битового бюджета, в то время как на компоненты  $Cb$  и  $Cr$  выделяется всего по 5%.

**Субвыборка в цветовых каналах.** На этом шаге из компонентов  $Cb$  и  $Cr$  удаляется каждый второй элемент по горизонтали и по вертикали. При декодировании, каждый пиксел заменяется на четыре пиксела с такими же значениями. Субвыборка позволяет значительно сократить количество обрабатываемых данных, повышая тем самым общую степень сжатия и скорость работы программы. Стоит отметить, что поскольку чувствительность к хроматическим составляющим значительно слабее, даже такой „грубый“ прием в большинстве случаев остается незаметным для человеческого восприятия.

**Вейвлетное преобразование.** Центральную роль в описываемом алгоритме играет вейвлетное преобразование. Задача вейвлетного преобразования заключается в выделении НЧ и ВЧ составляющих сигнала. Известно, что НЧ составляющая более ценна для человеческого восприятия нежели ВЧ. Таким образом, условно разделив информацию на „более важную“ и „менее важную“ можно закодировать первую с меньшими потерями чем вторую. В основном за счет этого и достигается сжатие. Реализуется вейвлетное преобразование путем применения к сигналу каскада из НЧ и ВЧ фильтров. Первые выделяют НЧ и подавляют ВЧ информацию, а вторые — наоборот.

**Кодирование коэффициентов.** Вейвлетные коэффициенты округляются до целых и кодируются методом SPECK [5]. Используя особенности структуры вейвлетных коэффициентов, алгоритм SPECK переупорядочивает их биты. При этом, первые биты будут нести наиболее важную информацию, в то время как последние — лишь незначительные, уточняющие детали. Такое упорядочение данных называют прогрессивным. Оно позволяет декодеру по мере поступления данных последовательно улучшать и уточнять реконструируемое изображение: вначале будут прорисованы основные цветовые и яркостные переходы, а уже затем второстепенные контуры и малозаметные детали. Прогрессивное упорядочение также позволяет задавать степень сжатия заранее: действительно, можно сохранить лишь требуемое количество первых битов закодированного изображения, а оставшийся „хвост“ просто отбросить, так как он несет сравнительно мало информации.

**Мультиплексирование каналов.** Итак, цветное изображение представляется в виде трех компонент  $Y$ ,  $Cb$  и  $Cr$ , причем каждая из них кодируется независимо друг от друга. Если изображение распаковывается и отображается локально, то вполне приемлемо сохранить все три потока по очереди. Если же изображение будет передаваться по сети и отображаться по мере загрузки, то сохранять потоки по отдельности нельзя, иначе пользователь будет наблюдать как вначале прорисовывается черно-белый вариант изображения, а затем уже изображение наполняется цветом. Для того чтобы изображение при передаче прорисовывалось равномерно по всем каналам, соответствующие битовые потоки необходимо равномерно перемешать (мультиплексировать) в один поток.

**Маркеры блоков.** Каждый закодированный блок отделяется от других блоков специальным маркером. Маркеры позволяют декодировать несколько блоков одновременно, перемешивать блоки в произвольном порядке, а так же значительно повышают устойчивость закодированных данных к повреждениям. Применение маркеров, однако, требует определенной осторожности: если в закодированных данных встретится соответствующая последовательность, то декодер по ошибке примет ее за конец блока. Для того чтобы исключить подобную двусмысленность необходимо использовать экранирование. В программе EPSILON реализован алгоритм экранирования COBS [6]. Он гарантирует, что даже в худшем случае рас-

ширение данных составит не более 0.4%.

**Двухпроходный алгоритм.** В программе реализованы две стратегии битового аллоцирования: CBR (Constant Bit-Rate) и VBR (Variable Bit-Rate). В первом случае битовый бюджет делится поровну между всеми блоками, а во втором — пропорционально их „сложности“. Блок называется сложным, если для его адекватной передачи требуется большое количество битов. К сложным блокам можно отнести фрагменты изображения со множеством мелких деталей и резких линий, а к простым — плавные переходы яркости и цвета. Если изображение неоднородно, то алгоритм CBR оказывается неэффективным: простые блоки зачастую даже не выбирают выделенный им битовый бюджет. В этом случае разумнее использовать стратегию VBR: битовый бюджет сэкономленный на простых блоках будет использован для кодирования сложных. Стратегия VBR, однако, проигрывает CBR в скорости: реализация VBR требует двух проходов по изображению.

### 3. ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА

EPSILON изначально проектировалась как параллельная система. Распараллеливание наиболее удобно на уровне блоков — то есть небольших квадратных фрагментов изображения. Поскольку блоки никак не связаны друг с другом, обрабатывать их можно параллельно. В программе реализовано два подхода к параллельной обработке. Первый подход основан на применении потоков стандарта POSIX. Этот метод отлично подходит для многопроцессорных компьютеров, а также для компьютеров с многоядерными процессорами. Второй подход основан на применении MPI (Message Passing Interface) — технологии которая очень широко используется на высокопроизводительных кластерах. Результаты испытаний программы приведены в приложении.

### БЛАГОДАРНОСТИ

Автор выражает свою благодарность Научно-исследовательскому вычислительному центру МГУ имени М.В.Ломоносова<sup>1</sup> за предоставление доступа к высокопроизводи-

---

<sup>1</sup> НИВЦ МГУ: <http://www.srcc.msu.su/>

тельными кластерам, а также лаборатории компьютерной графики и мультимедиа при факультете ВМиК МГУ<sup>2</sup>, за поддержку исследований в области сжатия изображений.

## ПРИЛОЖЕНИЕ А: РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

Испытания проводились на кластере ANT, конфигурация которого приведена в табл. 1. Тестовое изображение последовательно сжималось и распаковывалось с использованием различного числа процессоров. Результаты замеров приведены в табл. 2.

- 
1. *Симаков А.В.* Сжатие изображений при помощи вейвлетных преобразований // Вестник молодых ученых. Сер. Прикладная математика и механика. 2004. Вып. 4. С. 53–62.
  2. *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: Диалог-МИФИ, 2003.
  3. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб: БХВ-Петербург, 2002.
  4. *Wei J., Pickering M.R., Frater M.R., Arnold J.F., Boman J., Zeng W.* Boundary artifact reduction using odd tile lengths and the low pass. first convention (OTLPPF) // Proc. of SPIE. Appl. of Digital Image Proc. July 2001.
  5. *Asad I., Pearlman W.* An embedded and efficient low-complexity hierarchical image coder // Proc. of SPIE. Visual Comm. and Image Processing 99. 1999. V. 3653. P. 294–305.
  6. *Cheshire S., Baker M.* Consistent overhead byte stuffing // IEEE/ACM Transactions on Networking. 1999. V. 7, № 2, P. 159–172.

---

<sup>2</sup> Graphics & Media Lab: <http://graphics.cs.msu.ru/>

# EPSILON — PARALLEL IMAGE COMPRESSION SYSTEM

**A. Simakov**

This article describes parallel image compression system called EPSILON. The system is based on wavelet transform and industry standards for parallel computing. The program is successfully tested on several high-performance computing systems. EPSILON is an open-source project, so full source code is available for download<sup>a</sup>.

<sup>a</sup> <http://epsilon-project.sourceforge.net>

**Таблица 1.** Сжатие и распаковка полноцветного изображения размера 3072 × 2304 точек на кластере ANT

Кол-во процессов	Кодирование (сек.)	Декодирование (сек.)
2	10.365	8.531
3	4.594	4.264
4	3.258	2.972
5	2.501	2.171
6	2.008	1.784
7	1.698	1.559
8	1.563	1.417
9	1.399	1.297
10	1.251	1.199
11	1.148	1.160
12	1.091	1.140

**Таблица 2.** Конфигурация кластера ANT

Узлов	80
Ядер	160
Конфигурация узла	2xOpteron 2.2 ГГц, 4 Gb RAM
Коммуникационная сеть	Infiniband
Транспортная сеть	Gigabit Ethernet
Сервисная сеть	FastEthernet
Пиковая производительность	704 GFlops