Параллельный метод сжатия изображений для визуализации данных на массивно-параллельных вычислительных системах

О.В. Джосан, Н.Н. Попова

В работе рассмотрен метод параллельного сжатия изображений для системы визуализации результатов научных вычислений, проводимых на терафлопсных и петафлопсных вычислительных системах. Предложен метод, основанный на блочном кодировании изображений, использующий быстрый способ оценки визуальной сложности блоков изображения, что позволяет априорно оценить временные затраты на кодирование и осуществить разбиение блоков по процессорам для минимизации общего времени кодирования изображения. Получена в среднем десятикратная степень сжатия при качестве без визуальных потерь.

1. Введение

Одним из основных инструментов анализа результатов научных вычислений на массивно-параллельных вычислительных системах является их визуальное представление.

Визуализация результатов научных вычислений, проводимых на массивно-параллельных вычислительных системах (МПВС), обладает рядом особенностей. Получаемый в вычислительных экспериментах объем данных может быть очень большим. Например, при моделировании турбулентного горения на вычислительной системе Blue Gene/P для проекта FLASH с использованием 8000 четырехядерных вычислительных узлов генерируется 16Gb данных каждые 10-15 минут, общий объем данных эксперимента составляет 300Tb[1]. Объем данных, получаемых в задаче климатического моделирования, составляет 345Tb[2]. Объем данных, получаемых в результате проведения экспериментов на системе Blue Gene/P, установленной в МГУ, измеряется сотнями гигабайт. Система визуализации результатов научных вычислений на МПВС должна поддерживать возможность работы с данными большого объема.

В работе рассмотрен метод параллельного сжатия изображений и видео для системы визуализации результатов научных вычислений, проводимых на терафлопсных и петафлопсных вычислительных системах. Сжатие получившихся при визуализации данных является важным методом в системе визуализации, т.к. позволяет существенно сократить объем передаваемых данных. Возможно использование нескольких процессоров для сжатия, что позволяет осуществлять работу по сжатию видеопоследовательности в параллельном режиме.

В данной работе для сжатия изображений и видеопоследовательностей предложен метод кодирования, основанный на блочном кодировании изображения. При этом используется быстрый способ оценки сложности блоков изображения, что позволяет оценить временные затраты на кодирование и осуществить разбиение блоков по процессорам в зависимости от этого времени, чтобы минимизировать общее время кодирования кадра. Каждое изображение или кадр видеопоследовательности кодируется как последовательность блоков, при этом для каждого блока подбирается оптимальный способ его кодирования в зависимости от посчитанной визуальной сложности. Для кодирования используются палитровые методы кодирования, основанные на поиске оптимальной палитры для блока. При этом палитра, соответствующая блоку, может состоять из разного количества цветов в зависимости от визуальной сложности блока.

Применение блочного кодирования для метода параллельного сжатия изображений и видеопоследовательностей продиктовано необходимостью разделения по данным: таким образом, чтобы блоки изображения могли бы обрабатывается независимо на разных процессорах. Однако использование блочного кодирования чревато появлением блочных артефактов. Поэтому используемый размер блока должен быть небольшим.

2. Параллельный метод сжатия изображений

Блок-схема предложенного параллельного метода сжатия изображений для системы визуализации показана Рис.1. На первом шаге изображение разбивается на блоки. Размер используемого блока – 4x4 пикселя. Для каждого блока осуществляется оценка его визуальной сложности, основанная на частотных характеристиках области, к которой относится блок. На основе этой характеристики проводится распределение полученных блоков по процессорам для минимизации общего времени кодирования блока. Кодирование блоков осуществляется с помощью нахождения оптимальной палитры для блока.

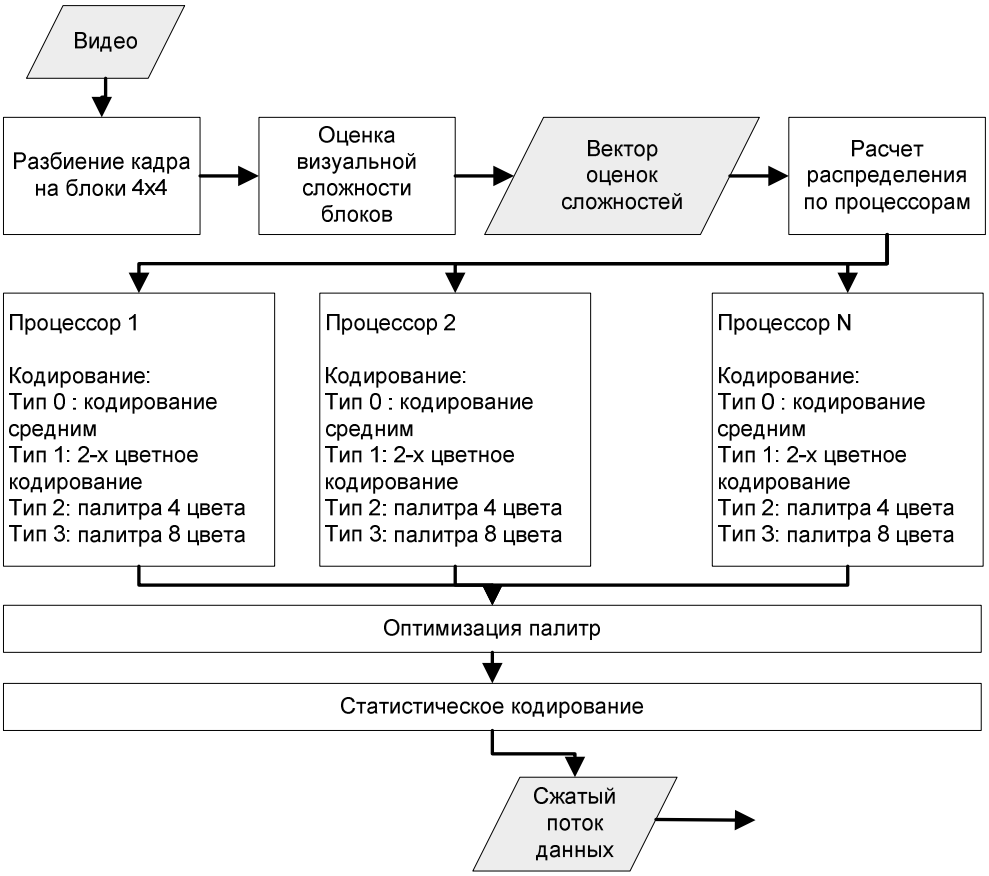


Рис.1 Блок-схема параллельного метода сжатия изображений для системы визуализации

Для оценки визуальной сложности блока строится интегрированная оценка, исходя из частотных характеристик блока. Оцениваются различные характеристики блока, которые показывают насколько близкие значения у соседних пикселей в блоке. Пиксели в блоке обрабатываются в формате YCbCr. Преобразование из RGB формата осуществляется по формулам[3]:

$$Y = 0.257R + 0.504G + 0.098B + 16$$
$$Cb = -0.148R - 0.291G + 0.439B + 128$$
$$Cr = 0.439R - 0.368G - 0.071B + 128$$

Обратное преобразование при декодировании осуществляется по формулам:

$$R = 1.164(Y - 16) + 1.596(Cr - 128)$$
$$G = 1.164(Y - 16) - 0.813(Cr - 128) -0.392(Cb - 128)$$
$$B = 1.164(Y - 16) + 2.017(Cb - 128)$$

3. Оценка визуальной сложности блока

Предложено оценивать визуальную сложность блоков с помощью следующего интегрального соотношения:

$$F = \frac{\sigma}{12}\left(\frac{D_x}{4} + \frac{D_Y}{4} + \frac{D_{d1}}{3} + \frac{D_{d2}}{3}\right) + \frac{\sigma}{2}\sum_{T\in\{Y,Cb,Cr\}}\left(A_T + B_T\right)$$

где $\sigma, D_X, D_Y, D_{d1}, D_{d2}, A_T, B_T$ - соотношения пикселей в блоке, рассчитанные с помощью соотношений, приведенных далее.

Рассчитывается дисперсия в блоке и оптимальные цвета для двухцветного кодирования:

$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i, \quad \sigma = \frac{1}{k}\sum_{i=1}^{k}(x - x_i)^2, \quad a = \bar{x} - \sigma\times\sqrt{\frac{q}{k-q}}, \quad b = \bar{x} + \sigma\times\sqrt{\frac{k-q}{q}}$$

где k – это количество пикселей в блоке, q – это количество пикселей в блоке, которые больше среднего значения в этом блоке. Проводится оценка количества и резкости границ в блоке следующим образом, где каждый коэффициет это дискретная производная в точке по выбранному направлению:

$$D_X = \sum_{\substack{k=-1..1\\ l=-1..2}} \left|I_Y(i+k+1, j+l) - I_Y(i+k, j+l)\right|$$

$$D_Y = \sum_{\substack{k=-1..2\\ l=-1..1}} \left|I_Y(i+k, j+l+1) - I_Y(i+k, j+l)\right|$$

$$D_{d1} = \sum_{\substack{k=-1..1\\ l=-1..1}} \left|I_Y(i+k+1, j+l+1) - I_Y(i+k, j+l)\right|$$

$$D_{d2} = \sum_{\substack{k=-1..1\\ l=-1..1}} \left|I_Y(i+k+1, j+l) - I_Y(i+k, j+l+1)\right|$$

Дополнительно для оценки сложности проводится расчет параметров частотности области следующим образом:

$$A_T = \frac{1}{4}\sum_{\substack{h\in\{-1,1\}\\ g\in\{-1,1\}}} \left|I_T(i, j) - I_T(i+h, j+g)\right|$$

$$B_T = \frac{1}{4}\left(\left|I_T(i, j-1) - I_T(i-1, j)\right| + \left|I_T(i-1, j) - I_T(i, j+1)\right| + \left|I_T(i, j+1) - I_T(i+1, j)\right| + \left|I_T(i+1, j) - I_T(i, j-1)\right|\right)$$

где T – это цветовой канал, I(i,j) – пиксель, относительные координаты которого в блоке (1,1).

Далее определяется, какому из заданных диапазонов значений принадлежит получившееся значение F. Номер диапазона соответствует номеру метода кодирования, который оптимален для блока. В текущей реализации количество возможных методов кодирования M=4. Однако это значение может быть расширено: возможно добавление и замена методов кодирования.

Для оптимизации времени вычислений предложен метод оптимального распределения блоков по вычислительным процессорам, основанный на оценке визуальной сложности блока. На каждом процессоре соответствующие блоки кодируются с помощью одного из 4-х методов: кодирование средним, двухцветное кодирование, кодирование с помощью 4-х цветной палитры, кодирование с помощью 8-х цветной палитры.

Для оптимизации распределения блоков по вычислительным узлам используется равномерное распределение блоков в зависимости от их сложности. Для этого блоки сортируются в порядке значений из визуальных сложностей, затем последовательно распределяются по процессорам.

4. Методы кодирования блоков

Рассмотрим более подробно предложенные методы кодирования блоков. Первый метод кодирования требует наименьшее количество бит для кодирования блока и имеет наименьшую вычислительную сложность, но при этом вносится много визуальных искажений. При использовании этого метода пиксели блока кодируются средним значение для пикселей этого блока. Это значение подсчитано на этапе оценки сложности блока. Таким образом, количество бит, требуемое для кодирования в данном случае, равно количеству бит, которым передается значение одного цвета. При этом самый младший бит не передается, а при декодировании восстанавливается нулем, поэтому в случае 256 цветов, требуется семь бит.

Следующий метод кодирования – это метод BTC, описанный в [6,7]. Основная идея этого метода состоит в том, что для кодирования блока используются два цвета. Эти два цвета определяются по формулам:

$$a = \bar{x} - \sigma \times \sqrt{\frac{q}{k-q}}, \quad b = \bar{x} + \sigma \times \sqrt{\frac{k-q}{q}}$$

где q – это количество пикселей в блоке, значение которых больше, чем среднее значение $\bar{x}$, σ – значение дисперсии.

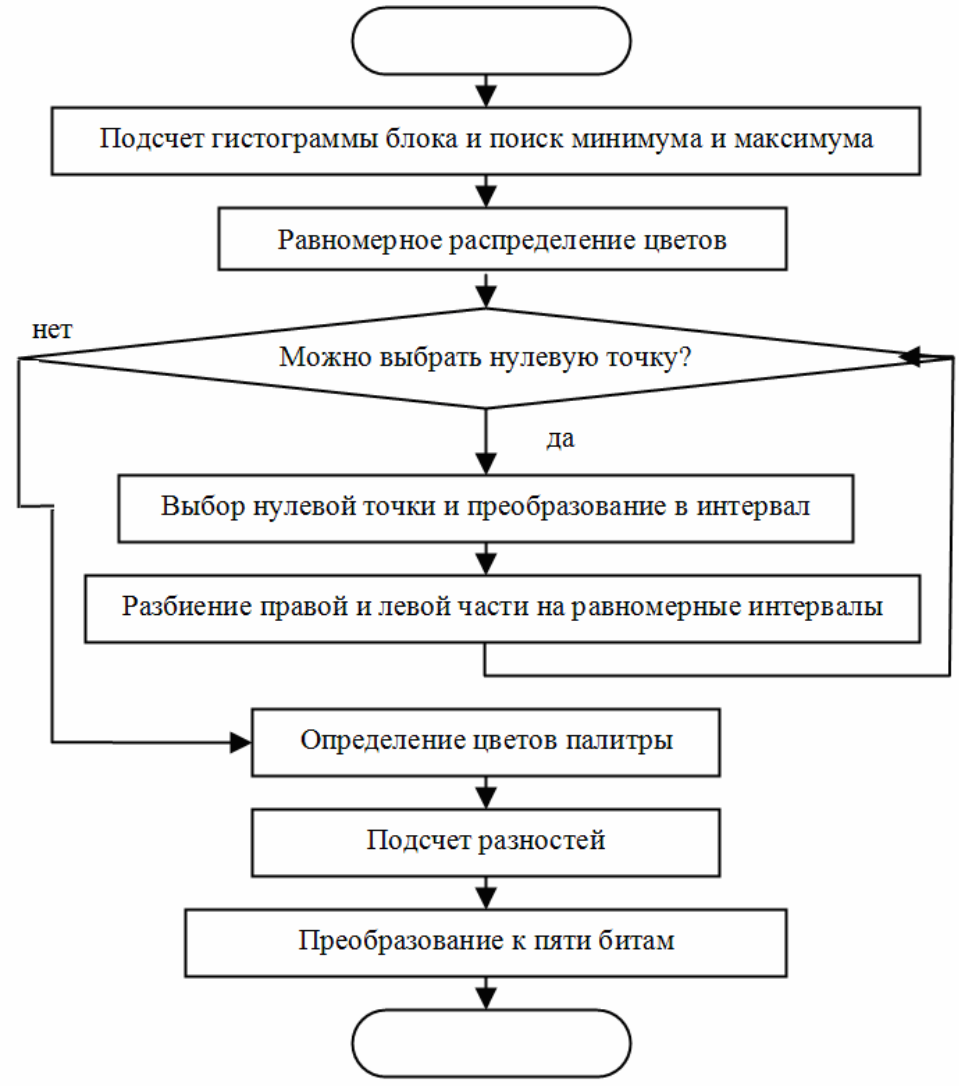


Рис.2 Методы определения оптимальной палитры для блока

При этом в качестве закодированной последовательности передается среднее значение $\bar{x}$, дисперсия σ и битовая карта для пикселей, где бит, соответствующий пикселю, равен единице, если значение пикселя больше, чем среднее значение для блока, и равно нулю иначе. При этом $\bar{x}$, σ уже посчитаны на шаге оценки сложности блока. Этот метод требует 28 бит для передачи блока: 16 бит на битовую карту, 8 бит для передачи среднего значения и 4 бита для передачи дисперсии. Этот метод используется для кодирования блоков с большей визуальной сложностью, чем у блоков, кодируемых первым методом.

Еще один метод, который используется для кодирования, - это определение оптимальной палитры. Для блока определяется оптимальная палитра и карта, в которой указаны номера цветов палитры, которые наиболее близки к цветам пикселей блока. Предложен метод определения оптимальной восьмицветной и четырехцветной палитры. Критерий оптимальности выбран следующий: минимизировать максимальную разность между цветами закодированного блока и реальными цветами блока. Для достижения этого критерия предложено минимизировать количество нулевых интервалов в гистограмме и цвета палитры распределять между оставшимися частями гистограммы.

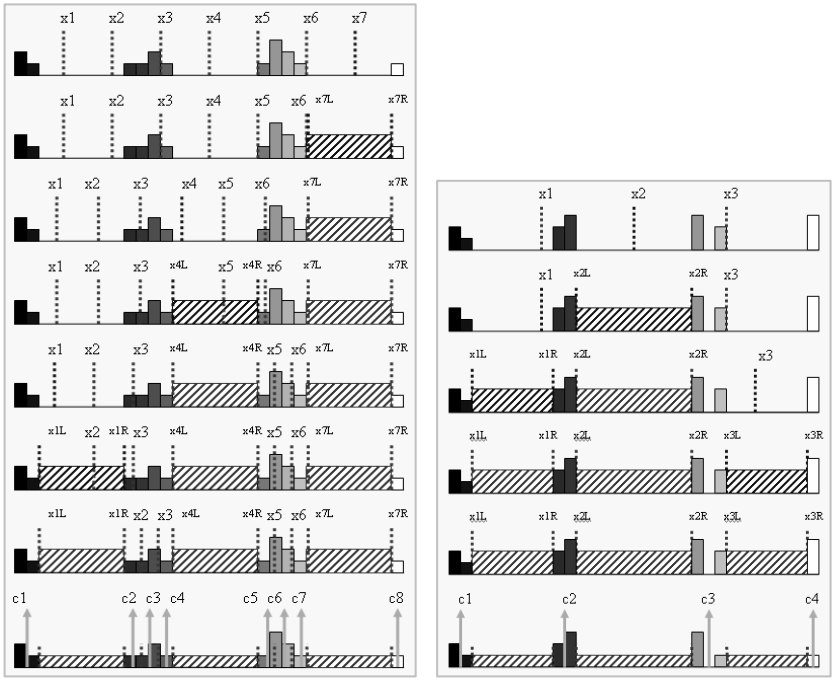
Схема этого метода кодирования показана на Рис.2 и процесс определения оптимальной палитры проиллюстрирован на Рис.3а для восьмицветной палитры, на Рис.3б для четырехцветной палитры. Палитра определяется по распределению цветов в исходной гистограмме.

Рассмотрим применение алгоритма построения оптимальной палитры из 8 цветов на примере. На первом шаге определяют гистограммы блока. Далее выбирают рабочий интервала [x0, x8], где x0 является минимальным ненулевым значением гистограммы, x8 является максимальным ненулевым значением гистограммы. На следующем шаге рабочий интервал разбивают на восемь интервалов одинаковой длины точками множества A = {x1, x2, x3, x4, x5, x6, x7}. Затем проверяют, принадлежат какие-либо точки xi к нулевому интервалу и, если такие точки имеются, выбирают точку множества A с наименьшим индексом, принадлежащей самому длинному нулевому интервалу, с последующим исключением xi из множества A. Далее точку xi преобразуют в две точки – xiL и xiR – наиболее близкие к xi слева и справа ненулевые значения в гистограмме. Слева и справа от этих двух новых точек соответственно получаются два новых рабочих интервала гистограммы. Оставшиеся точки множества A распределяют в этих двух интервалах, проводя равномерное разбиение интервалов этими точками: точки с номерами меньше i распределяют в левом интервале, а точки с большими чем i номерами индексов – в правом интервале. Далее снова выбирают и преобразовывают в интервал точку из множества A, которая попала в наибольший нулевой интервал, если такая точка может быть выбрана. Этот процесс повторяется до исчерпания точек во множестве A, попадающих в нулевые интервалы на гистограмме. Далее определяют оптимальную палитру как середины восьми получившихся интервалов.

Получившуюся цветовую палитру преобразуют во множество разностей между соседними цветами, при этом палитру изменяют так, чтобы все разности стали меньше 32. Т.е. в худшем случае получается равномерное распределение цветов палитры по гистограмме. Однако для большинства изображений и видеопоследовательностей значение получившейся статистики показывают, что соседние цвета находятся недалеко друг от друга. На Рис.4а показана статистика разностей цветов в палитрах блоков изображения, показанного на Рис.4б. Как видно из графика большинство значений разностей изначально не превышают значения в 32 градации, поэтому преобразование палитр затронет только незначительное количество блоков, качество которых не повлияет на общее визуальное качество картинки.

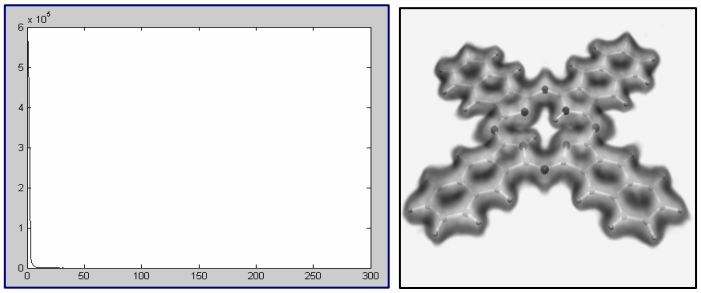
На первом шаге для изменения палитры добавляют фиктивные цвета, если это можно сделать. В отношении фиктивного цвета сначала вычисляют, является ли число требующихся фиктивных цветов меньше или равно числу имеющихся фиктивных цветов (если количество использованных цветов в палитре меньше восьми), если число фиктивных цветов достаточно, то их равномерно распределяют по длинным интервалам, если это не достаточно, то разбивают рабочий интервал гистограммы на восемь одинаковых интервалов и рассматривают середины этих интервалов в качестве палитры. Рис.5а иллюстрирует предложенную идею добавления фиктивных цветов. После добавления фиктивных цветов, в случае если остались разности

больше 32 градаций, проводится смещение цветов. Перестановка цветов проиллюстрирована на Рис.5б.



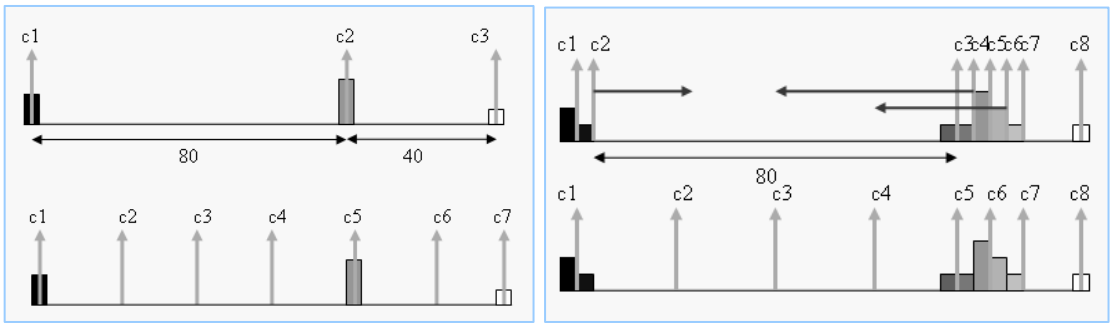
а) б)

Рис.3 Пример расчета оптимальной палитры для блока: а) 8 цветов; б) 4 цвета



а) б)

Рис.**4** статистика разностей в цветах палитр



а) б)

Рис.5 а) Добавление фиктивных цветов в палитру; б) Перестановка цветов в палитре

После определения цветовой палитры вычисляют цветовую карту, в которой номер цвета на палитре, ближайшей к реальному значению пикселя, является значением пикселя в цветовой карте. Аналогичный алгоритм применяется для определения четырехцветной палитры. Кодирование восьмью цветами требует 91 бит, кодирование четырьмя цветами – 64 бита. Эта длина закодированного пакета включает в себя два бита, которые определяют тип кодирования для блока, по которому на стороне декодера будет осуществлено восстановление изображения.

Статистическое кодирование подразумевает сжатие без потерь потока информации. Для этого применяется сворачивание цепочек нулей RLE [8].

5. Эффективность предложенного метода сжатия

Для измерения количественных характеристик эффективности предложенного метода сжатия рассмотрим следующий модельный эксперимент по моделированию молекулярных переключателей с помощью кода CPMD. В рассматриваемой модели использовалось 82 атома, 129 орбиталей. Объем данных эксперимента составил около 200Gb. Для счета использовалось 512 процессоров, при этом время расчета составляет около суток. Для эксперимента использовались первые 1024 файла исходных данных, представленных в наборе файлов формата .cube, каждый из которых описывает положение молекулы в момент времени.

На тестовом примере проанализирована эффективность алгоритма сжатия. Размер получаемого изображения по одному файлу до применения алгоритма сжатия составлял от 120К для изображения 200x200 до порядка 2М для изображения 800x800. На Рис.6 представлен график степени сжатия для нескольких кадров рассматриваемой видеопоследовательности. График показывает степень сжатия для изображения размером 200x200 для качества без визуальных потерь с использованием предложенного многопоточного метода сжатия. Из графика видно, что степень сжатия меняться. В среднем степень сжатия равна 10. Таким образом, исходный размер данных вычислительного эксперимента, используемых при данном модельном эксперименте, составлял порядка 31 гигабайта. Размер полученной в этом эксперименте видеопоследовательности до применения сжатия составил 123М, размер полученной видео последовательности после применения алгоритма сжатия составил около 12М.

Рис. 5 Степень сжатия для модельного эксперимента

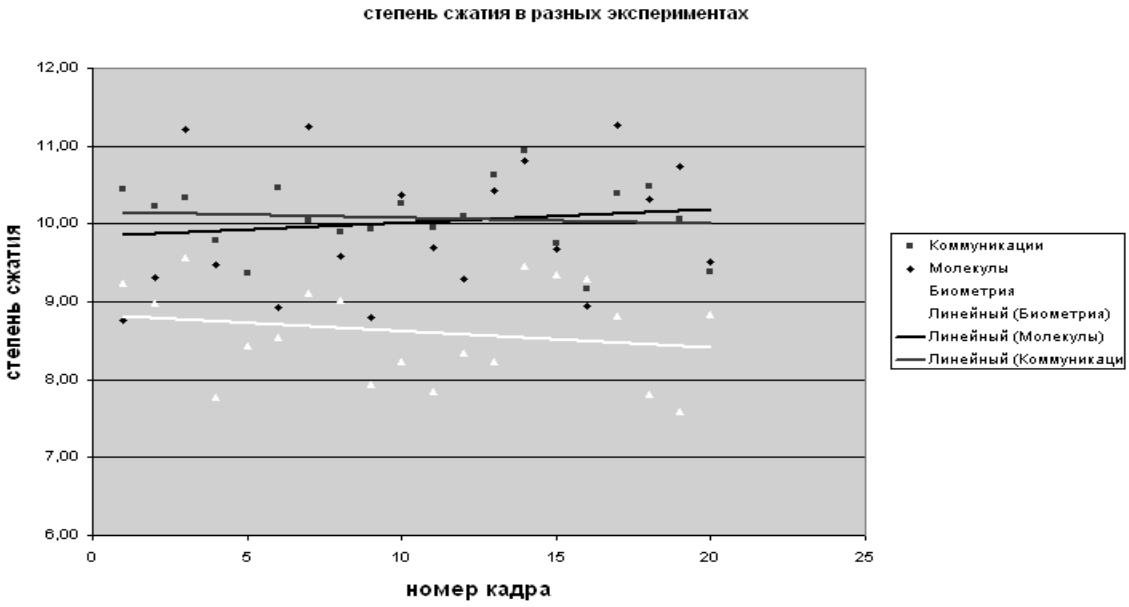


Рис. 6 Сравнение полученных степеней сжатия в различных экспериментах

На Рис.6 показано сравнение степеней сжатия для применения визуализации в реальных данных прикладных экспериментов.

6. Заключение

Разработан специализированный метод многопоточного сжатия изображений, который основан на блочном кодировании кадра и использует специализированную функцию оценки визуальной сложности блока перед распределением блоков для кодирования на нескольких процессорах. Предложена соответствующая функция оценки визуальной сложности блоков, основанная на частотных характеристиках. Предложенный метод сжатия изображений позволяет получить существенное улучшение качества сжатого изображения по сравнению со стандартными методами сжатия. Метод учитывает особенности входного изображения, а именно, тот факт, что сжатию подвергаются синтезированные изображения, которые включают в себя большие однородные области фона, объекты с ярко выраженными границами и высокочастотные области с большим количеством деталей. Эксперименты показали, что предложенный метод позволяет сжимать изображения с точностью максимум до 4 градаций по сравнению с 20 градациями, которые дает стандартный mpeg2 кодек при той же степени сжатия. Получено в среднем десятикратная степень сжатия при качестве без визуальных потерь.

Работа выполнена при поддержке грантов РФФИ 08-07-12081, 08-07-00445-а, 09-07-12068.

Литература

1. T. Peterka, R. B. Ross, H.-W. Shen, K.-L. Ma, W. Kendall, and H. Yu, "Parallel Visualization on Leadership Computing Resources," Preprint ANL/MCS-P1656-0709, July 2009.
2. T. Peterka, R. Ross, H. Yu, K.-L. Ma, W. Kendall, and J. Huang, "Assessing and Improving Large Scale Parallel Volume Rendering on the IBM Blue Gene/P," Preprint ANL/MCS-P1554-1008, October 2008
3. Keith Jack , YCbCr to RGB Considerations , http://www.intersil.com/data/an/AN9717.pdf
4. Мишуровский М.Н. Джосан О.В. Рычагов М.Н. Способ компактного представления цветных цифровых изображений, предназначенный для применения в системах цифровой обра-

ботки видеосигналов в реальном времени // тезисы конференции “Телевидение: передача и обработка изображений”, Россия, Санкт-Петербург, 2009

5. Джосан О.В., Мишуровский М.Н. Анализ методов сжатия цифровых цветных изображений без визуальных потерь // тезисы конференции “Телевидение: передача и обработка изображений”, Россия, Санкт-Петербург, 2008
6. Zeng, B. Wang, Q. Neuvo, Y. , BTC image coding using two-dimensional median filter roots // Circuits and Systems, , IEEE International Sympoisum, vol.1, pp. 400-403, 1991
7. Zeng, B.; Neuvo, Y., Interpolative BTC image coding with vector quantization // Communications, IEEE Transactions on Volume 41, Issue 10, Oct. 1993, pp.1436 – 1438
8. Arturo San Emeterio Campos, Run Length Encoding, http://www.arturocampos.com/ac_rle.html