Выделяют следующие уровни архитектуры грид:

Базовый уровень - содержит различные ресурсы, такие как компью-теры, устройства хранения, сети, и др.

Связывающий уровень - определяет коммуникационные протоколы и протоколы аутентификации, обеспечивая передачу данных между ресурсами базового уровня. Он основан на стеке протоколов TCP/IP.

Ресурсный уровень - реализует протоколы взаимодействия с ресурсами РВС и их управления.

Коллективный уровень - отвечает за глобальную интеграцию различных наборов ресурсов и может включать в себя службы каталогов; службы совместного выделения, планирования и распределения ресурсов; службы мониторинга и диагностики ресурсов; службы репликации данных;

Прикладной уровень - инструментарий для работы с грид и пользовательские приложения, исполняемые в среде ВО. Они могут использовать ресурсы, находящиеся на любых нижних слоях архитектуры Грид.

Протоколы и интерфейсы являются открытыми, универсальными и стандартными. Грид технология позволяют использовать ресурсы на максимальном объёме с минимальными затратами.

Литература

1. Бовбель Е.И., Паршин В.В Грид технология в системах автоматического распознавания речи - Зарубежная радиоэлектроника Успехи современной радиоэлектроники, 2008, №4, с 49-65.

2. Lippman R.P., Review of neural networks for speech recognition, Neural Computation, 2001, vol 1, no 1, p 1 38.

А.П. Михеев, М.А. Кудрина

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ОПТИМИЗАЦИИ АЛГОРИТМА ФРАКТАЛЬНОГО СЖАТИЯ С ПОМОЩЬЮ ПАРАЛЛЕЛИЗМА

(Самарский университет)

Введение

Одним из методов сжатия изображений является фрактальное сжатие. С помощью данного алгоритма можно добиться хорошей степень сжатия изображений, при слабозаметных для человека потерях в качестве изображения.

Понятия «фрактал» (fractus – состоящий из фрагментов, лат.) были предложены математиком Б. Мандельбротом в 1975 г. для обозначения нерегулярных, но самоподобных структур.

Одним из основных свойств фракталов является самоподобие. В самом простом случае небольшая часть фрактала содержит информацию обо всём фрактале [1].

Фрактальное сжатие базируется на поиске этого самоподобия между частями сжимаемого изображения. Вместо пиксельной информации сохраняются позиции областей и их преобразования, через которые воссоздается исходное изображение.

Общая схема алгоритма

Общая схема алгоритма представляет следующее:

1) исходное изображение разбивается на ранговые области и доменные области;

2) для каждой ранговой области подбирается пара «домен - преобразование домена», такая, что расстояние между ранговым блоком и результатом преобразования доменного блока минимальное из всех возможных;

3) если расстояние между ранговой областью и подобранной парой больше заданного ε (критерия погрешности) и размер ранговой области не минимален, то ранговая область дробится на области меньшего размера. Если такие области есть, то алгоритм переходит к шагу 1, иначе к шагу 4;

4) для каждой ранговой области в файл записывается выбранное преобразование и позиция выбранной доменной области.

Для реальной реализации алгоритма вводятся следующие ограничения:

а) ранговые (r_i) и доменные (d_i) области имеют форму квадрата;

б) ранговые области r_i не пересекаются;

в) преобразование доменной области w_i – состоит из: сжатия блока d_i до размера блока r_i; одного из 7 аффинных преобразований, либо отсутствия такового (A_i): повороты на 0º, 90º, 180º, 270º и отражения относительно горизонтали, вертикали, главной диагонали, побочной диагонали; корректировки точек блока по формуле $r_i(x,y) = u_i \cdot A_i(d_i)(x,y) + v_i$, где u_i и v_i – коэффициенты преобразования яркости.

В файл результата компрессии для каждого рангового блока заносится информация:

- позиция доменного блока d_i;
- коэффициенты преобразования u_i и v_i;
- размер блока;
- аффинное преобразование.

Позиция рангового блока кодируется положением записи в файле результата.

Коэффициенты преобразования яркости u_i и v_i для блока r_i и сжатого до размеров блока r_i преобразованного доменного блока d'_i вычисляются из условия минимизации функции расстояния δ [2]:

$$\sum_{x=1}^{N}\sum_{y=1}^{N}(u_i \cdot d'_i(x,y) - r_i(x,y) + v_i)^2, \quad (1)$$

где N – длина стороны рангового блока,

$$u_i = \frac{a_i}{b_i},$$

$$v_i = \bar{r}_i - u_i \cdot \bar{d}'_i,$$

$$a_i = \sum_{x=1}^{N}\sum_{y=1}^{N}\left(d'_i(x,y) - \bar{d}'_i\right)\cdot\left(r_i(x,y) - \bar{r}_i\right),$$

$$b_i = \sum_{x=1}^{N}\sum_{y=1}^{N}\left(d'_i(x,y) - \bar{d}'_i\right)^2,$$

$$\bar{d}'_i = \frac{1}{N^2}\cdot\sum_{x=1}^{N}\sum_{y=1}^{N} d'_i(x,y)\,,$$

$$\bar{r}_i = \frac{1}{N^2}\cdot\sum_{x=1}^{N}\sum_{y=1}^{N} r_i(x,y).$$

Оптимизация

Основной проблемой стандартного алгоритма фрактального сжатия является время его выполнения. Поиск пары «домен – преобразование» подразумевает полный перебор таких пар для каждого рангового блока.

Метод классификации подразумевает разбиение всего объема доменных и ранговых блоков по классам в зависимости от эвристического критерия похожести блоков. В этом случае, при поиске домена ранговый блок сравнивается не со всеми возможными доменными блоками в изображении, а только с доменами, имеющими соответствующий класс [3].

Метод эталонного блока подразумевает сопоставление ранговых блоков и пар «домен-преобразование» не друг с другом, а с неким эталонным блоком. Ранговому блоку сопоставляется пара, для которой разность между расстоянием пары до эталона и рангового блока до эталона минимальна по модулю. Таким образом, мы избегаем необходимости вычислять СКО между каждым ранговым и каждым доменным блоками, единожды вычислив СКО между доменами и эталонным блоком [4].

Параллелизм

Основными особенностями модели параллельного программирования являются более высокая производительность программ, применение специальных приемов программирования и, как следствие, более высокая трудоемкость программирования, проблемы с переносимостью программ [5].

В алгоритме фрактального сжатия целесообразно параллельно выполнять сопоставление ранговых и доменных блоков. На обработку каждой тройки ранговый «блок - доменный блок – преобразование» выделяется отдельный поток. Результат сопоставления заносится в индивидуальную для каждой тройки ячейку, поэтому конфликты с доступом к данным отсутствуют.

Вычислительный эксперимент

Для проведения вычислительного эксперимента было использовано изображение Chrome.bmp (256x256). Параметры сжатия: ε – 10, количество классов – 5, количество бит на коэффициент u_i – 7, количество бит на коэффициент v_i –

9. Вычисления в параллельных методах выполнялось с помощью системы CUDA на GPU NVIDIA GeForce 840M, в непараллельных – на CPU Intel Core I5-4200M.

График зависимости времени сжатия от метода оптимизации представлен на рисунке 1.

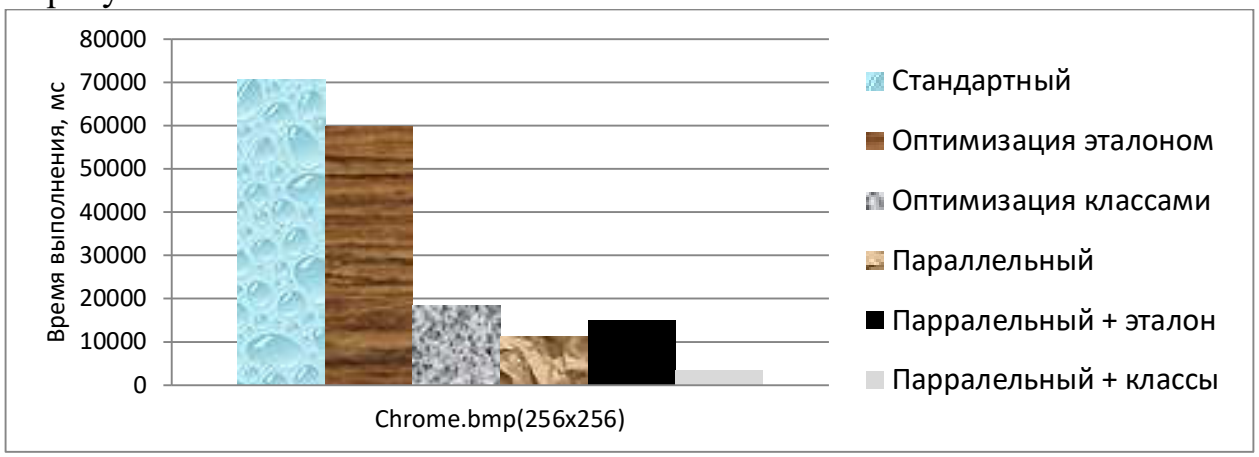

Рисунок 1 – График зависимости времени выполнения алгоритма от метода оптимизации для изображения Chrome.bmp

По результатам вычислительного эксперимента видно, что наибольшее ускорение работе алгоритма дает метод классификации с применением параллелизма на этапе сопоставления. Падение в качестве изображение при количестве классов равном 5 пренебрежимо мало. Исходное изображение и результат сжатия методом классификации представлены на рисунке 2.

a) b)

Рисунок 2 – Исходное изображение и результат сжатия, где (a) – исходное изображение , (b) – сжатое методом классификации.

Литература

1. Денисюк А.А., Полупанов А.А. Фрактальное сжатие изображений // Перспективные информационные технологии и интеллектуальные системы. – 2006. – №4. – ст.24-29.

2. Попов, Е. А. Классический алгоритм фрактального сжатия изображений [Текст] / Е. А. Попов, А. В. Холодков // Вестник Алтайской государствен-

ной педагогической академии. Сер.: Естественные и точные науки. - 2011. - Вып. 7. - С. 42-46 : табл. - Библиогр.: с. 46 - ISSN 2218-4767

3. Ансон, Л. Фрактальное сжатие изображения [Текст] // Л. Ансон, М. Барнсли. - Мир ПК, 1992, № 4, с. 52 – 58.

4. Сахибназарова В.Б. Использование эталонного метода для увеличения скорости фрактального сжатия изображения // Технические и математические науки. Студенческий научный форум. – 2018. – ст.75-80

5. Немюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем // БХВ-Петербург, 2002 г. – 397с.

U.S. Otajonov

MONITORING THE WORK OF THE CENTRAL PROCESSOR

(Tashkent university of information technologies
named after Muhammad al-Khwarizmi)

The central processor is the main and most important element of the system. Thanks to all the tasks associated with data transfer, command execution, logical and arithmetic operations. The main characteristics of the CPU are: clock speed, performance, power consumption, standards of the lithographic process used in production (for microprocessors) and architecture.

The early CPUs were created in the form of unique components for unique, and even unique, computer systems. Later, from an expensive way of developing processors designed to perform one single or several highly specialized programs, computer manufacturers switched to mass production of typical classes of multi-purpose processor devices. The trend towards standardization of computer components began in the era of rapid development of semiconductor elements, mainframes and minicomputers, and with the advent of integrated circuits, it has become even more popular. The creation of microcircuits made it possible to further increase the complexity of the CPU while reducing their physical dimensions. Standardization and miniaturization of processors led to a deep penetration of digital devices based on them into everyday life. Modern processors can be found not only in high-tech devices such as computers, but also in cars, calculators, mobile phones and even in children's toys. Most often they are represented by microcontrollers, where, in addition to the computing device, additional components are located on the chip (program and data memory, interfaces, input / output ports, timers, etc.). Modern computing capabilities of the microcontroller are comparable with the processors of personal computers a decade ago, and often even significantly surpass their performance.

As you know, if a computer is operated in adverse conditions or in a dusty room, then over time it starts to work slower and slower as the condition of certain elements in its structure deteriorates.

Thus, among the most common causes of malfunctioning of modern computers are the following: