

« СРАВНИТЕЛЬНЫЙ АНАЛИЗ РАБОТЫ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА МАСШТАБИРОВАНИЯ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ ДЛЯ МНОГОЯДЕРНЫХ СРУ»

Выполнил Бугулов М.Р.

Научный руководитель: Мирошников А.С.

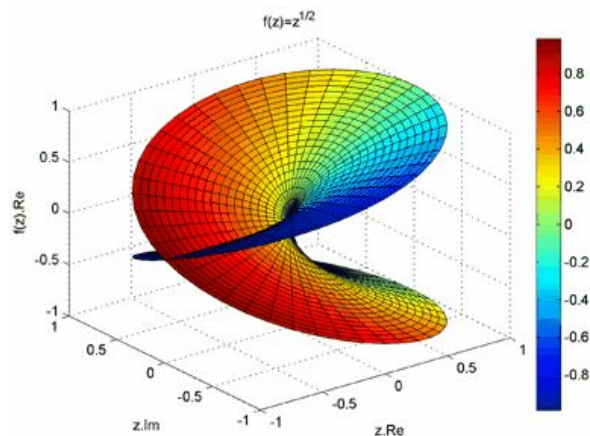
Цели и задачи

- Цель: минимизация времени и сравнительный анализ работы параллельного алгоритма масштабирования изображений.

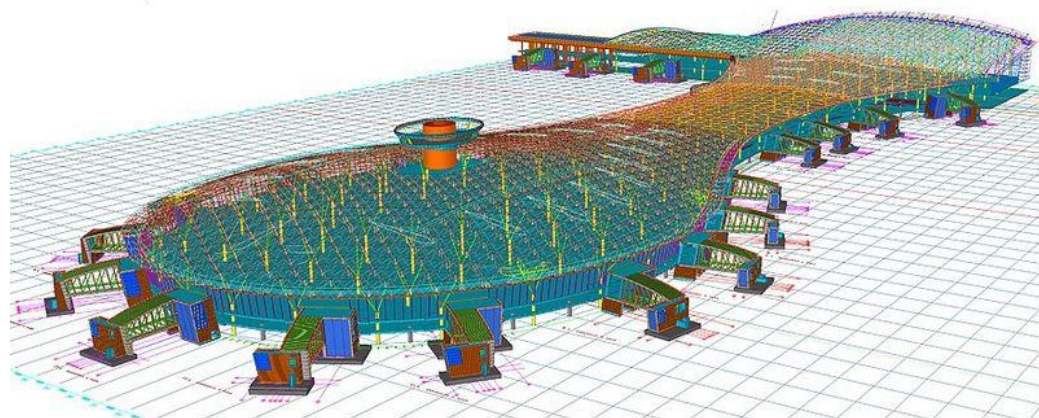
Задачи:

- Проведение аналитического обзора по данной теме;
- Выбор математической модели для минимизации времени масштабирования;
- Разработка параллельного алгоритма масштабирования;
- Программная реализация построенного алгоритма;
- Экспериментальная проверка эффективности выбранной математической модели и разработанного программного комплекса.

Актуальность



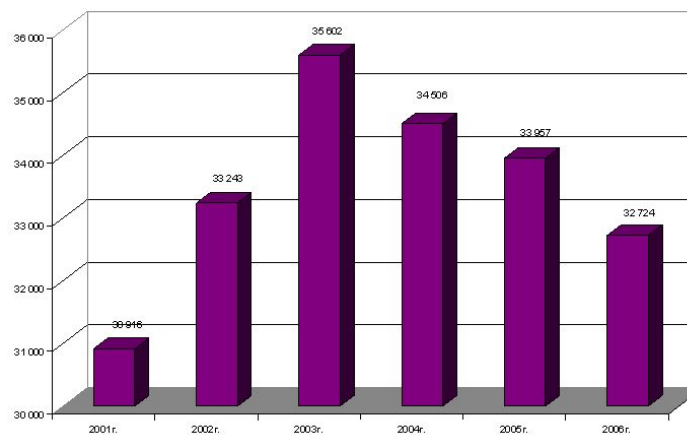
Научная графика



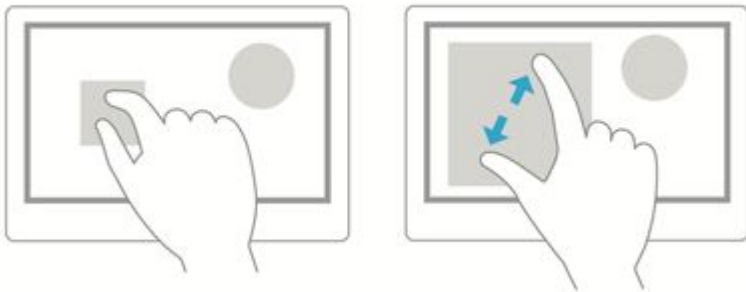
Конструкторская графика



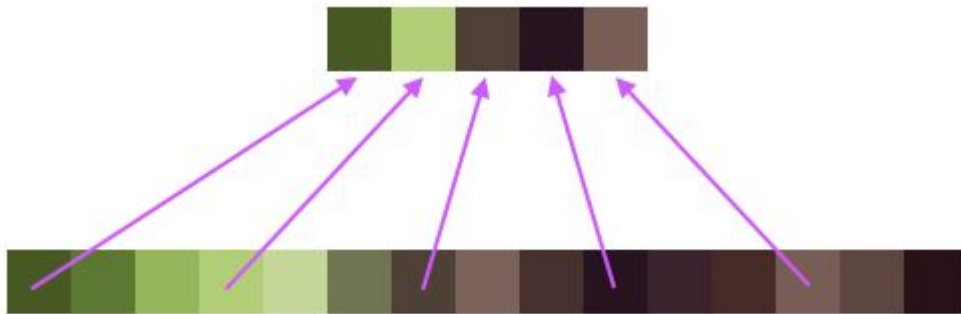
Художественная графика



Деловая графика



Описание алгоритма

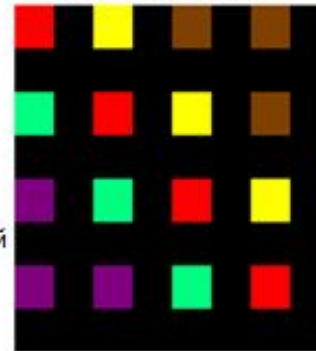


Принцип работы алгоритма при уменьшении изображения. Показано уменьшение в 3 раза.

w1 - 4 пикселя
h1 - 4 пикселя

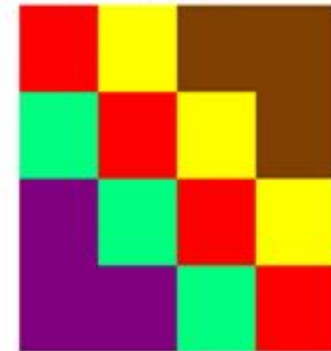


w2 - 8 пикселей



h2 - 8 пикселей

Готово



Принцип работы алгоритма при увеличении изображения. Показано увеличение в 2 раза

Пример масштабирования алгоритмом Nearest Neighbor

Уменьшенное изображение

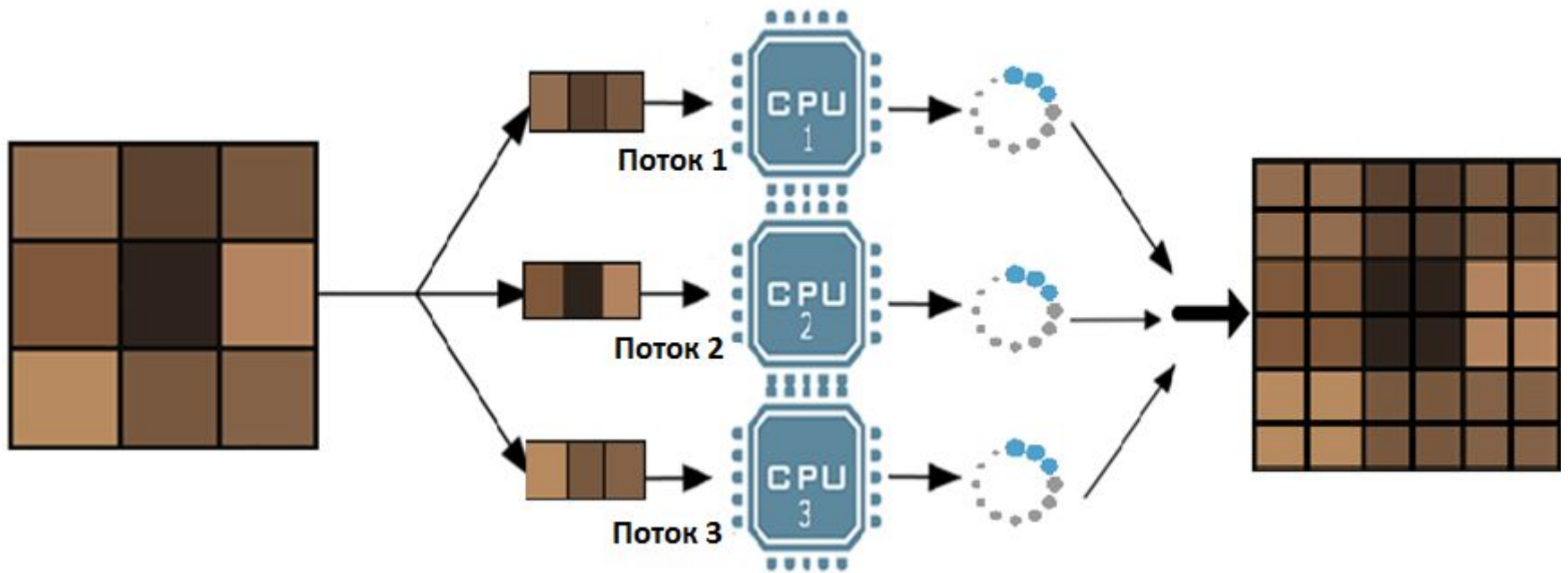


Увеличенное изображение



Оригинальное изображение

Параллельная реализация алгоритма



МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

$$\begin{cases} T = c + f \cdot x + \frac{d(m, n)}{x} \rightarrow \min \\ 1 \leq x \leq P, \text{ целое} \end{cases} \quad (1)$$

c – среднее время на организацию вычислений;

f – среднее время на организацию работы одного потока;

x – количество потоков, задействованных в параллельном алгоритме для масштабирования изображений;

$d(m, n)$ – время масштабирования изображения одним потоком;

P – максимальное количество активных потоков, поддерживаемых CPU;

n – высота результирующего изображения;

m – ширина результирующего изображения.

Расчет параметра $d(m,n)$

	Стоимость	Повторы
Инициализация массива b	C_1	1
For ($i = 0; i < n; i++$) {	C_2	n
For ($j = 0; j < m; j++$)	C_3	$n*m$
{процедура обработки пикселя $b[i,j]$ }	C_4	$n*m$
}		



$$d(m, n) = C_1 + C_2 * n + C_3 * n * m + C_4 * n * m = (C_3 + C_4) * n * m + C_2 * n + C_1$$

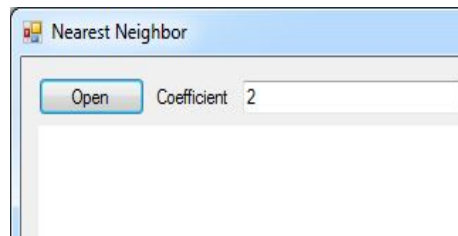
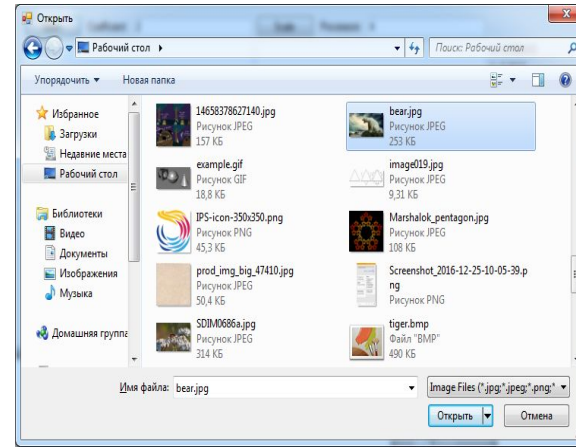
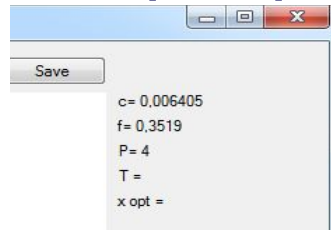
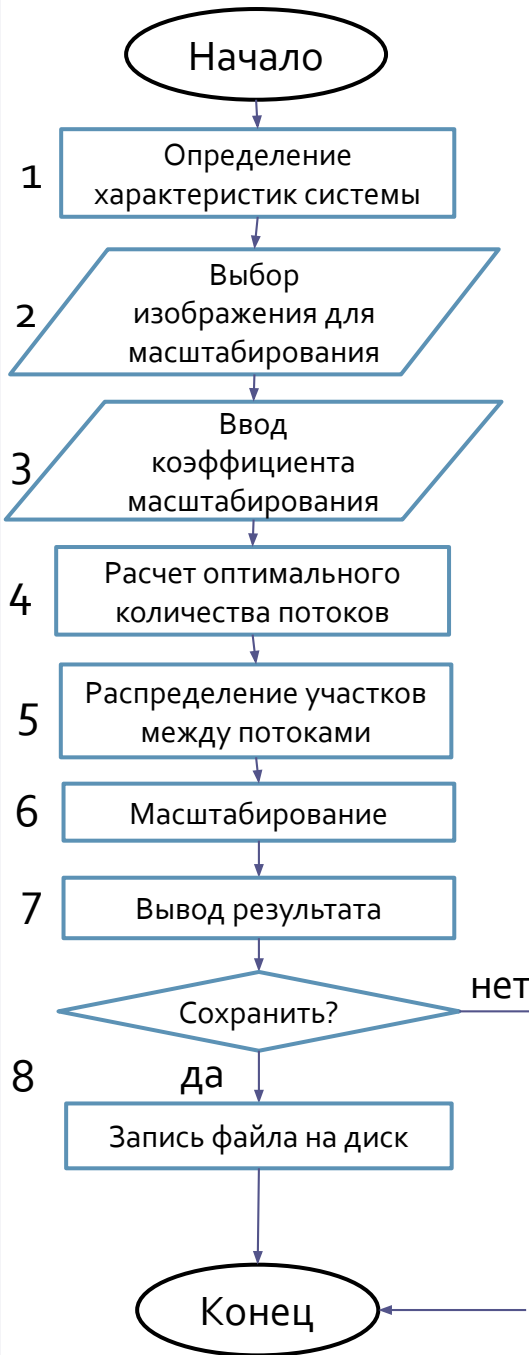
Обозначив $k_1 = C_3 + C_4$, $k_2 = C_2$ и $k_3 = C_1$,

время работы алгоритма можно записать:

$$d(m, n) = k_1 * n * m + k_2 * n + k_3.$$

(2)

Программный комплекс



Пример расчета оптимального количества пс
 Дано изображение 1900x1900 px, P=4
 $d(m,n) = 1,84E-06 * (n * m) + 1,284E-07 * n + 9,8E-05$
 $c = 6,361E-03$ секунд
 $f = 3,459E-01$ секунд.

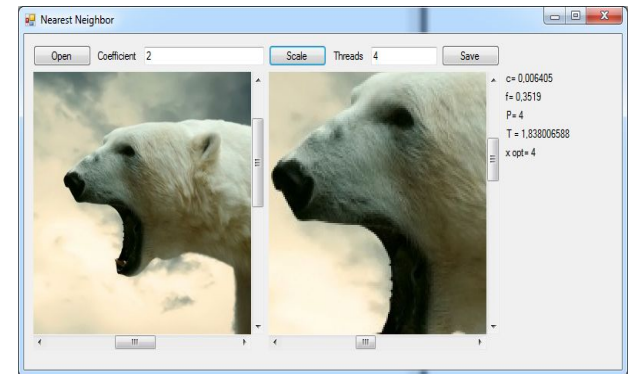
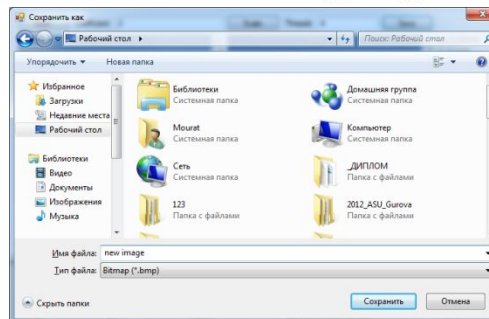
```

Thread t2 = new Thread(new ParameterizedThreadStart(scale2));
Thread t3 = new Thread(new ParameterizedThreadStart(scale3));
Thread t4 = new Thread(new ParameterizedThreadStart(scale4));
Thread t5 = new Thread(new ParameterizedThreadStart(scale5));
  
```

$$T'(x) = 0 + f + d * \left(-\frac{1}{x^2}\right) = 0$$

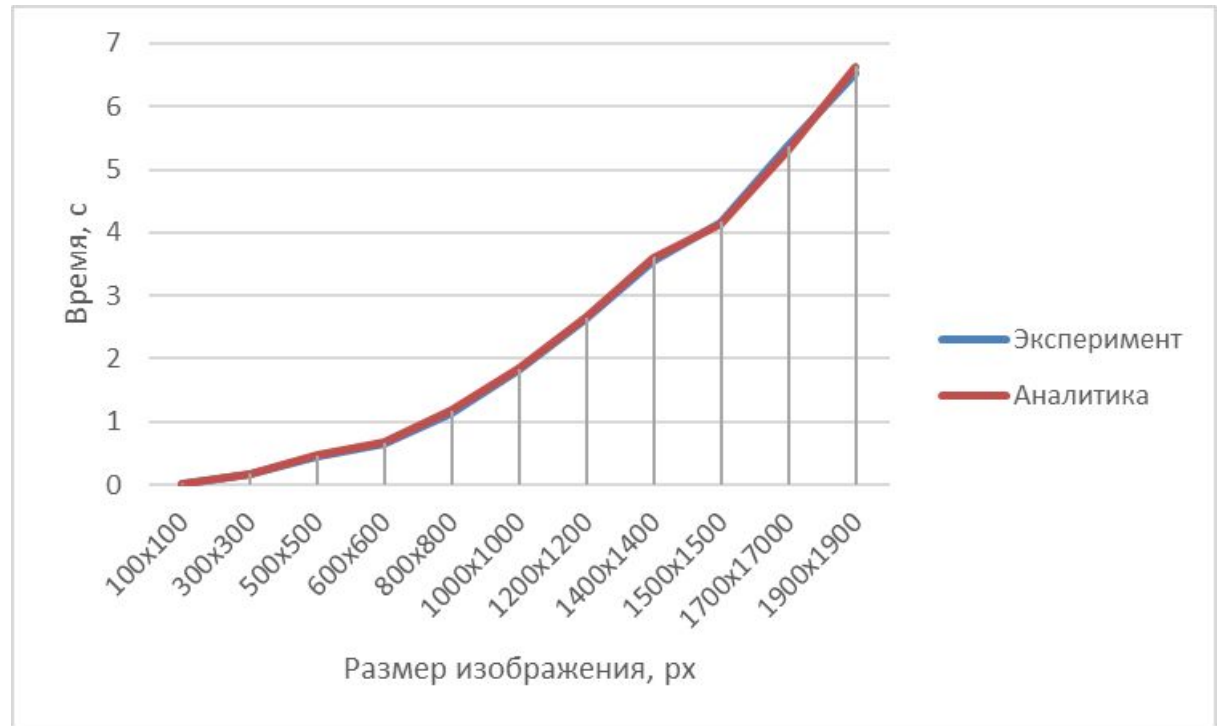
$$x = \sqrt{\frac{d}{f}} = \sqrt{\frac{1,84E-06(1900+1900) + 1,284E-07 * 1900 + 9,8E-05}{3,459E-01}} = 4,36$$

Т.к. $x > P$, то оптимальное количество потоков будет 4. $T(x) = 4.36$ с.]



Эксперимент 1

Размер изображения, px	Эксперимент, с	Аналитика, с
100x100	0,02087974	0,018733
300x300	0,16296232	0,165857
500x500	0,44846666	0,460105
600x600	0,64442029	0,662401
800x800	1,1347506	1,177335
1000x1000	1,80897122	1,839394
1200x1200	2,63456706	2,648576
1400x1400	3,55396282	3,604883
1500x1500	4,16899386	4,138208
1700x1700	5,37627342	5,315201
1900x1900	6,53036216	6,639319



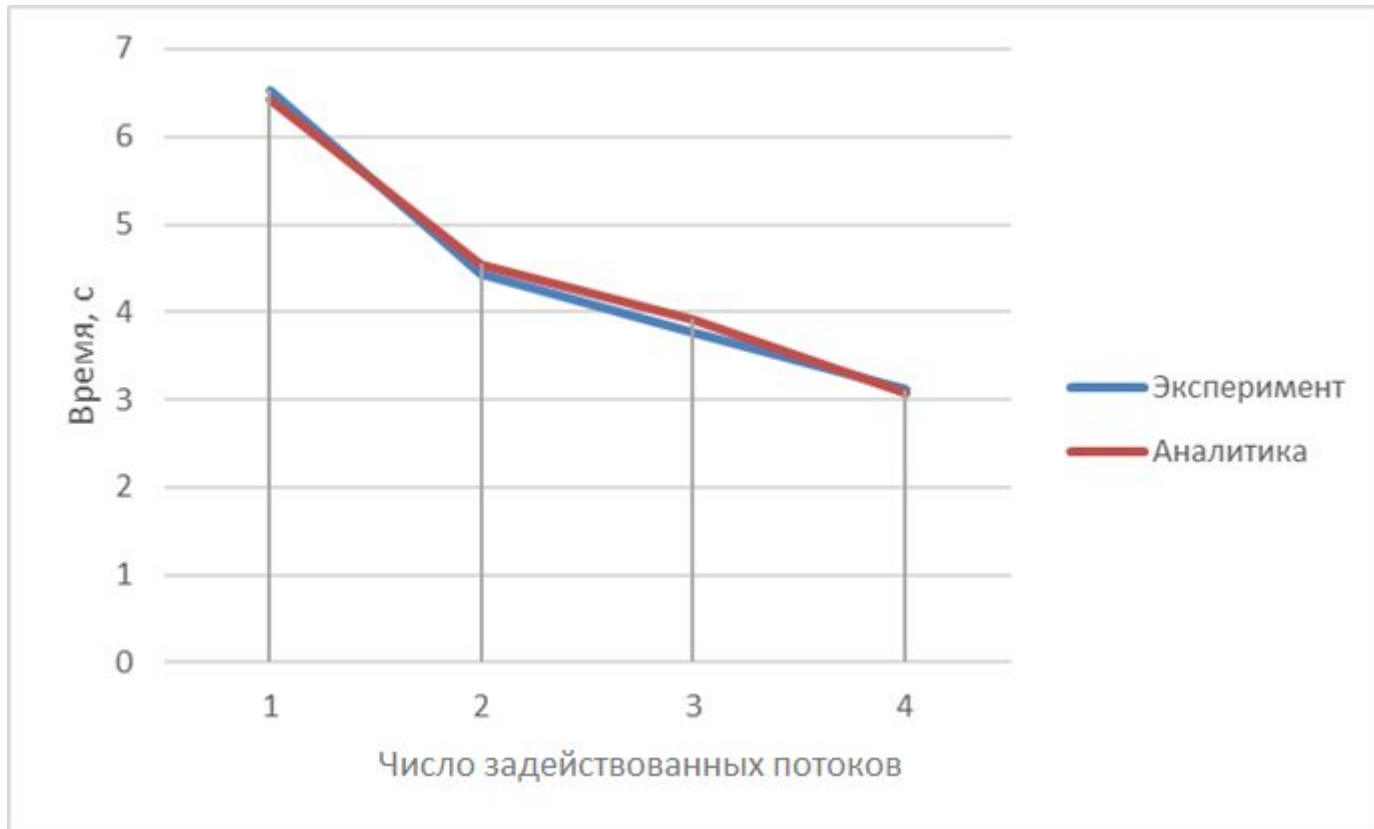
$$k_1=1,84E-06$$

$$k_2=1,284E-07$$

$$k_3=9,8E-05$$

Эксперимент 2

Р	Эксперимент	Аналитика
1	6,53036216	6,639319
2	4,43177562	4,5483575
3	3,86364146	3,9181778
4	3,11221826	3,0392697



Эксперимент 3

Размер\ потоки	1	4
100x100	0,02088	0,008942
300x300	0,162962	0,079999
500x500	0,448467	0,219866
600x600	0,64442	0,313288
800x800	1,134751	0,57777
1000x1000	1,80897	0,881983
1200x1200	2,634567	1,284812
1400x1400	3,553962	1,753722
1500x1500	4,168994	1,968062
1700x17000	5,376273	2,473851
1900x1900	6,530362	3,112218

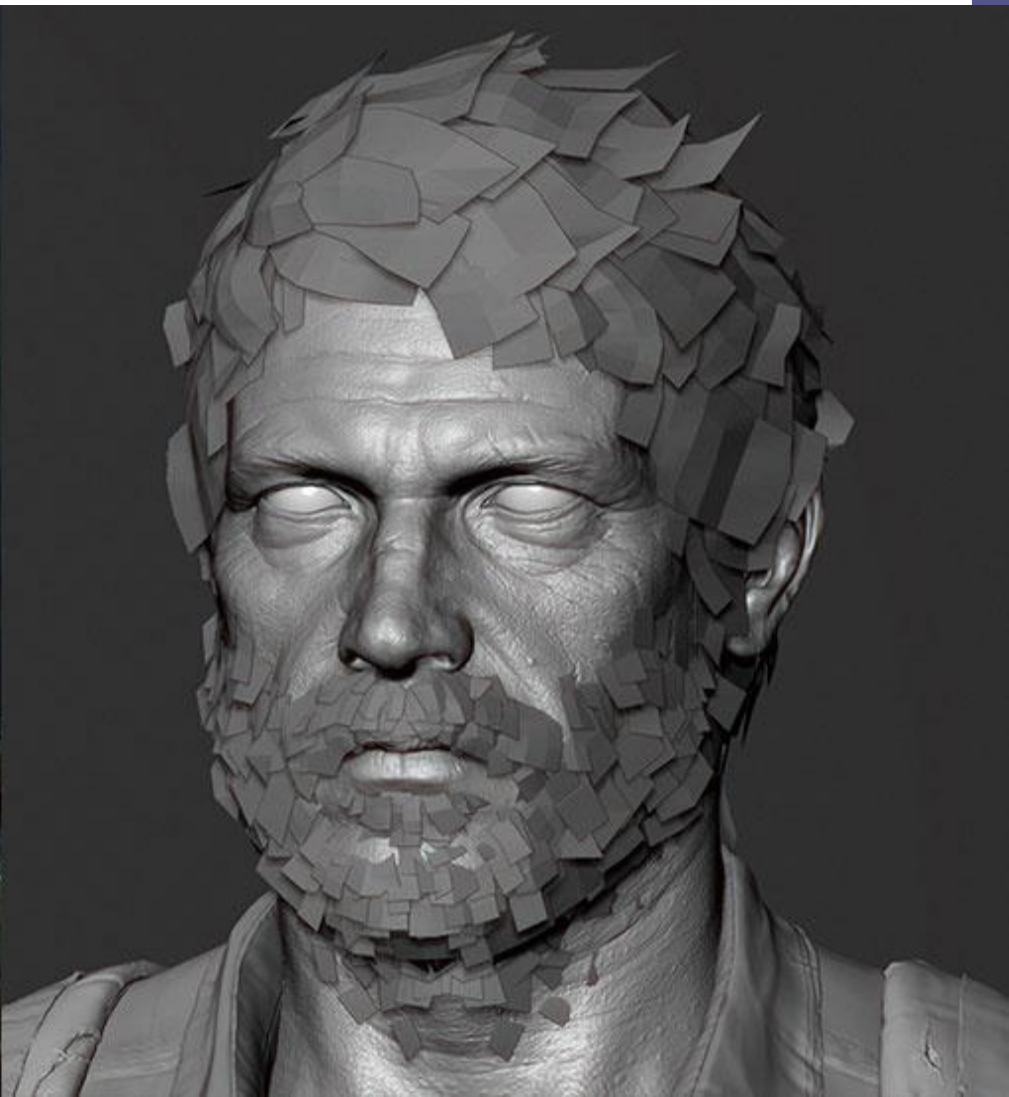


Заключение

- 1. Был проведен аналитический обзор предметной области;
- 2. Была выбрана математическая модель, минимизирующая время работы алгоритма масштабирования;
- 3. Был разработан параллельный алгоритм масштабирования;
- 4. Создан программный комплекс, реализующий параллельный алгоритм Nearest Neighbor;
- 5. Был проведен сравнительный анализ параллельного алгоритма с его последовательной реализацией.

Программный комплекс

Одна из важных частей при создании 3D сцен – текстуры моделей



Текстуры позволяют имитировать
жизненные сцены.

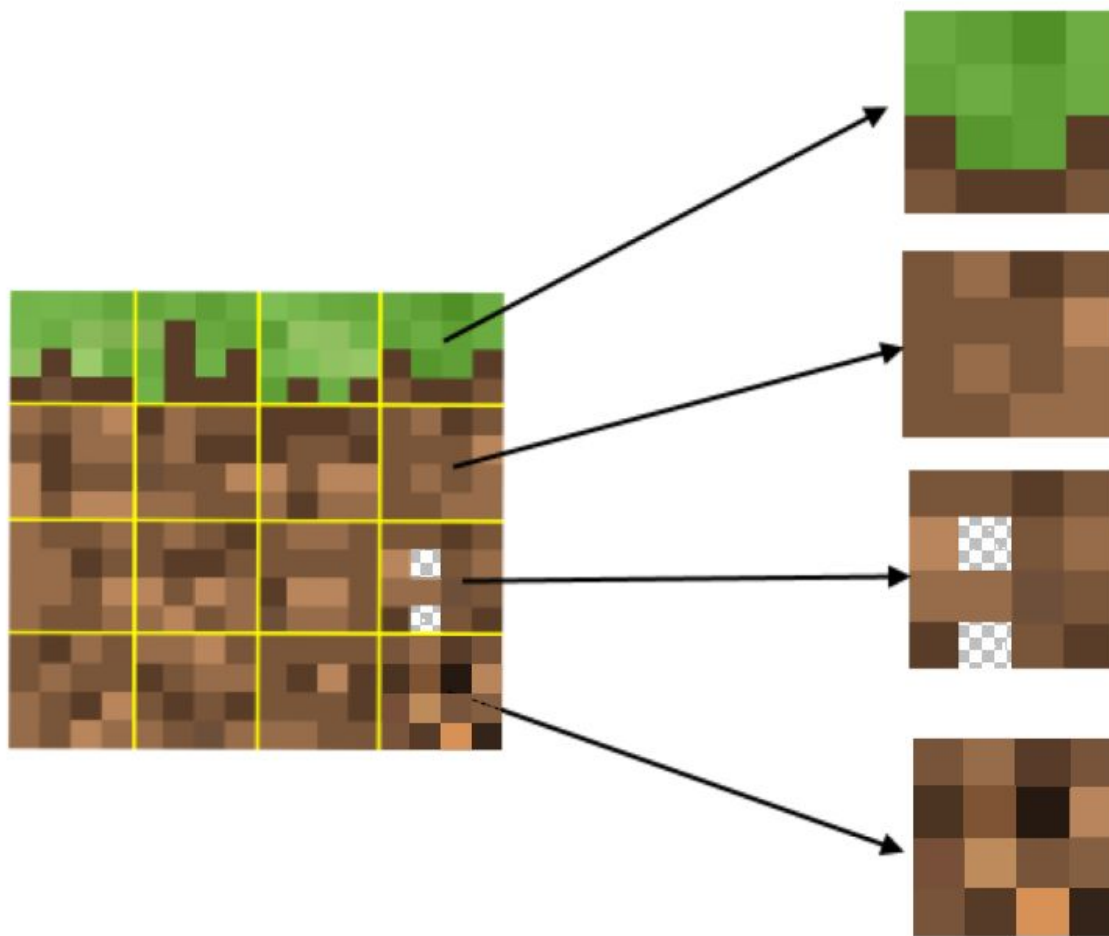


Grand
Theft
Auto

Группа алгоритмов DXT

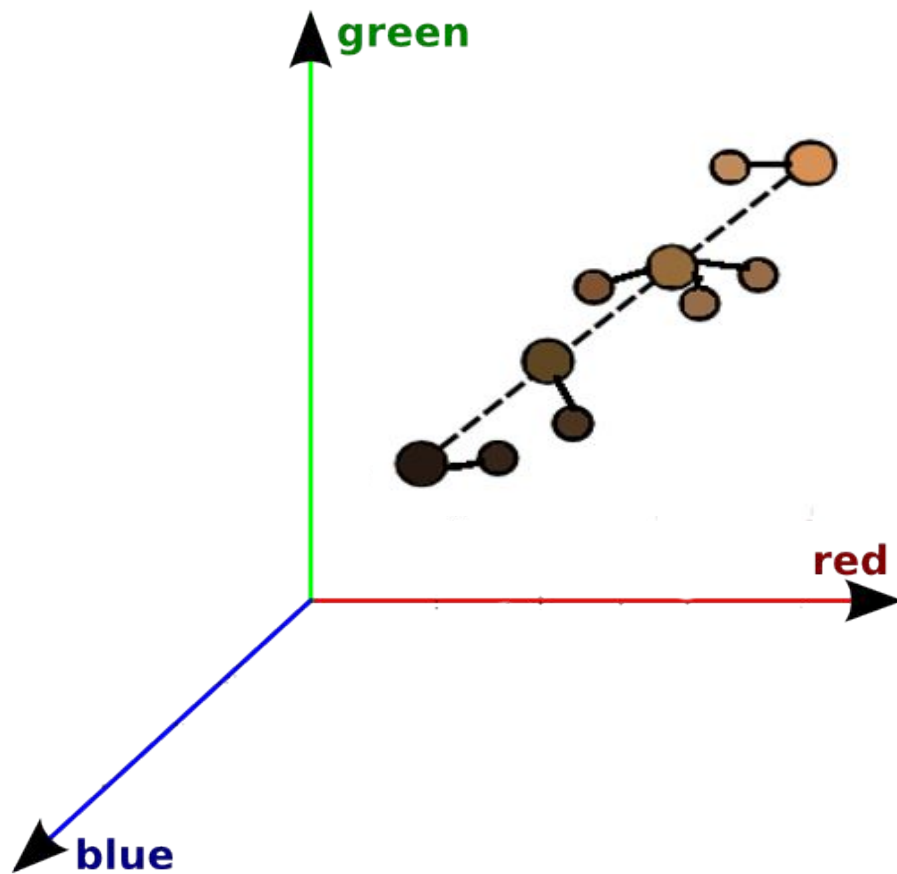
- DXT₁ – сжатие текстур с однобитным альфа-каналом
- DXT₃ – сжатие текстур с четырёхбитным альфа-каналом, содержащим произвольные значения
- DXT₂ – аналогично DXT₃, но с предумножением цвета на альфа-канал
- DXT₅ – сжатие текстур с восьмибитным альфа-каналом, содержащим табличные значения
- DXT₄ – аналогично DXT₅, но с предумножением цвета на альфа канал

Разбиение на блоци



Пример обработки блока

Обрабатываемый блок
4x4 пикселя

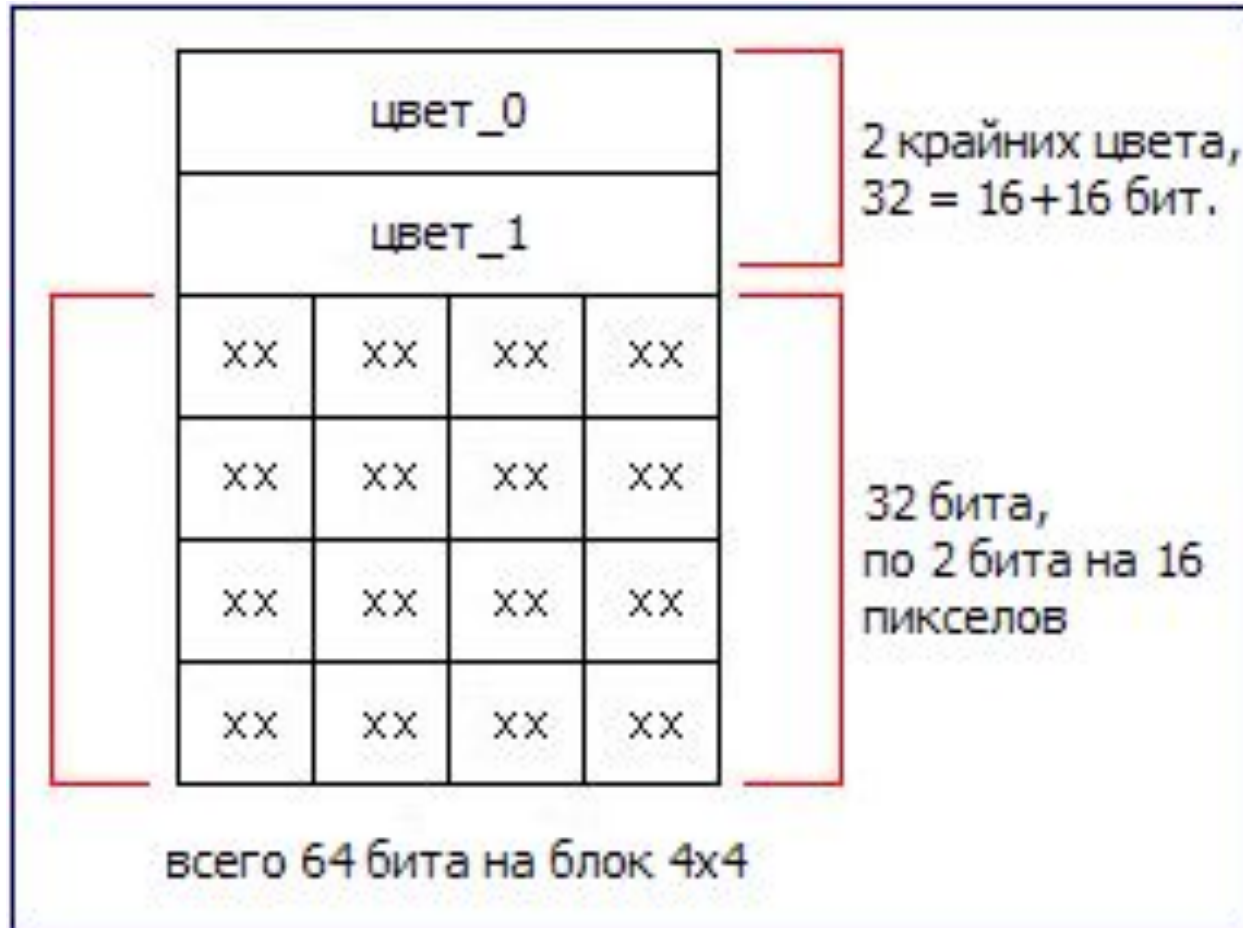


Формирование таблицы

				ЦВЕТ 01	10	10		10		
				Производные		11	10		00	01
				ЦВЕТ 10	ЦВЕТ 11 (альфа)	10	01	10	10	
				ЦВЕТ 00	10		11	01	00	

$$C_2 = \frac{1}{2}C_0 + \frac{1}{2}C_1$$

Структура сжатого блока



Результат DXT1



Несжатое изображение 1020* 960 px
3.7 MB



DXT1 сжатие 1020* 960 px
0,46 MB



DXT - сжатие с потерями



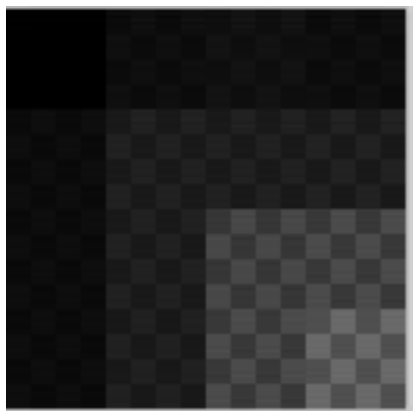
Увеличенный участок
несжатого изображения



Увеличенный участок
восстановленного изображения

DXT3

Преобразование альфа канала



Исходное изображение

255	240	240	240
240	220	220	220
240	220	180	180
240	220	180	150

Альфа исходного
изображения



Восстановленное изображение

15	14	14	14
14	13	13	13
14	13	11	11
14	13	11	9

Альфа в сжатом
блоке

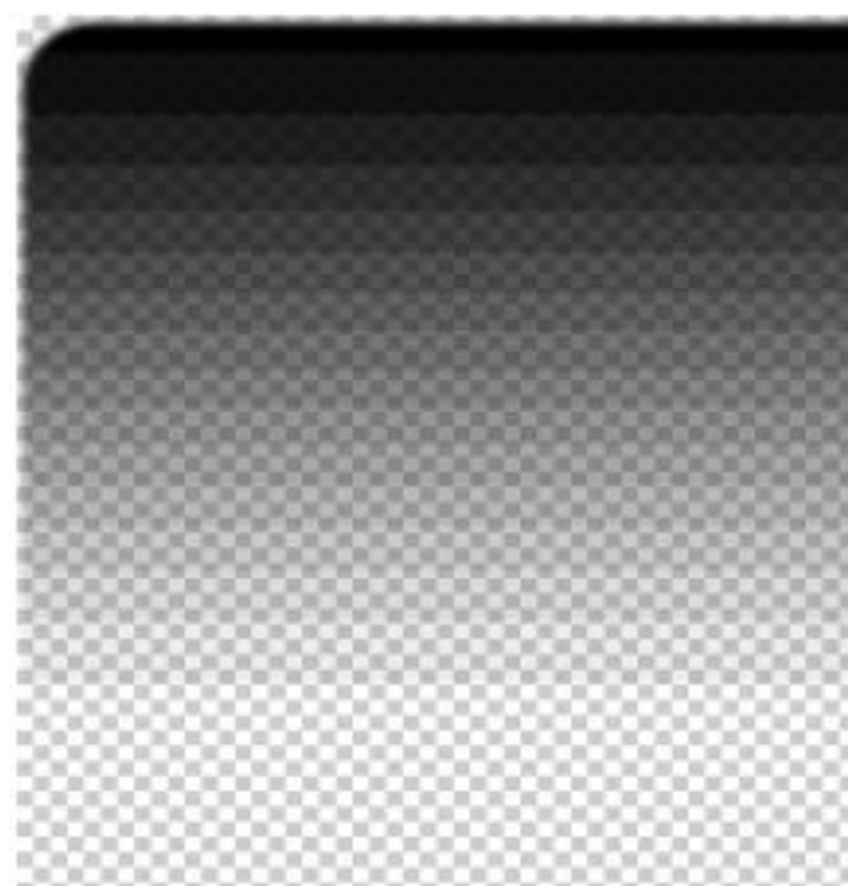
255	238	238	238
238	221	221	221
238	221	187	187
238	221	187	153

Альфа в восстановленном
блоке

Результат DXT3



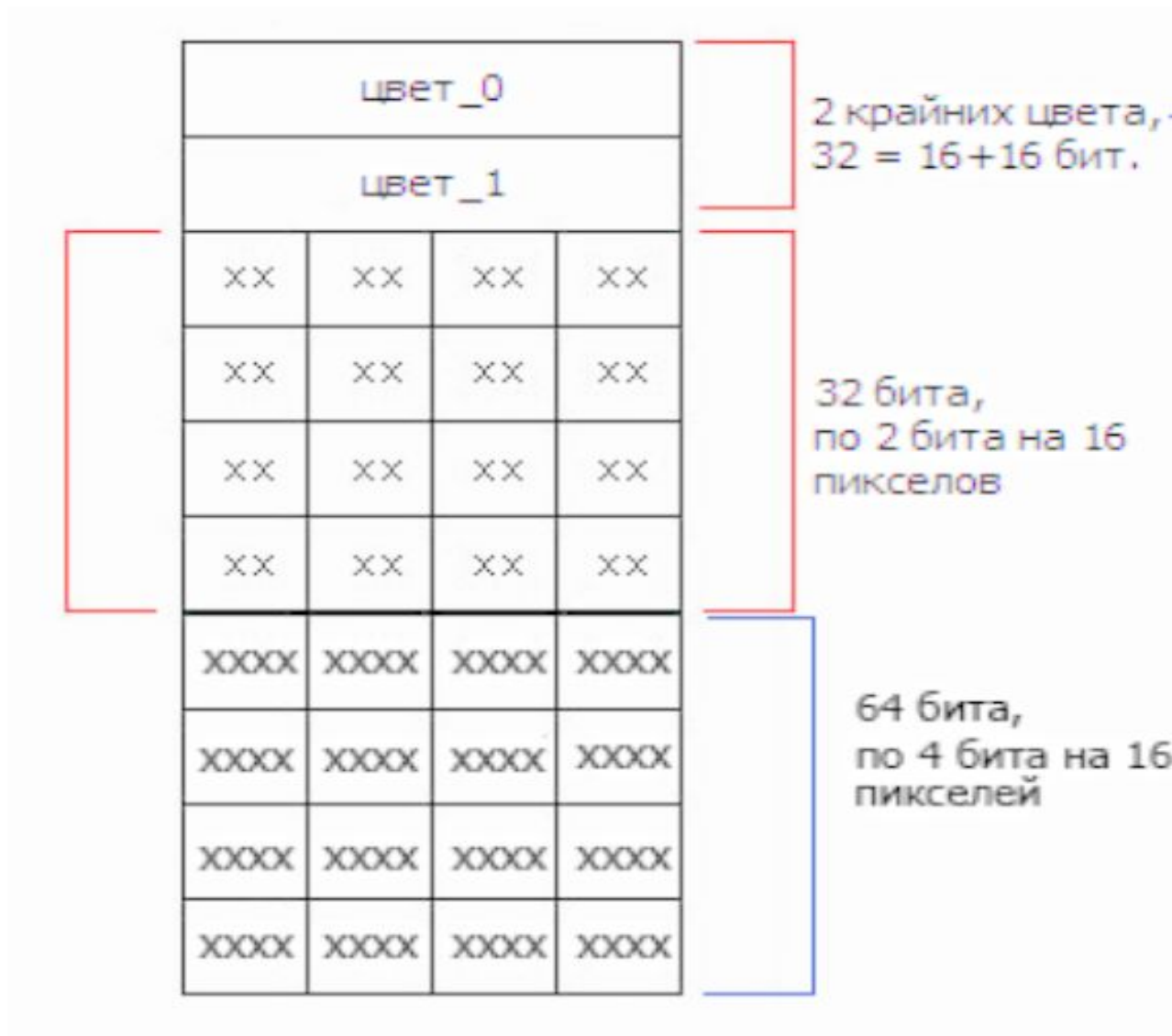
Несжатое изображение
1,47 MB



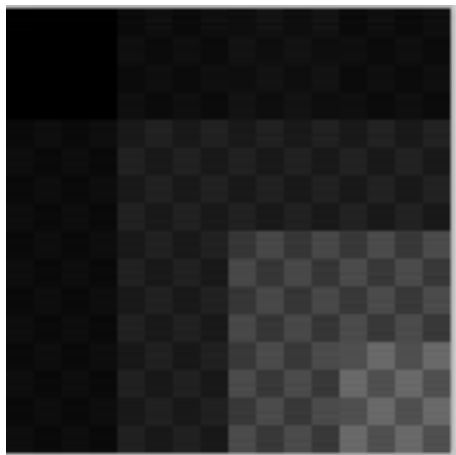
Сжатое изображение
0,35 MB



Структура DXT3 блока



DXT5. Пример

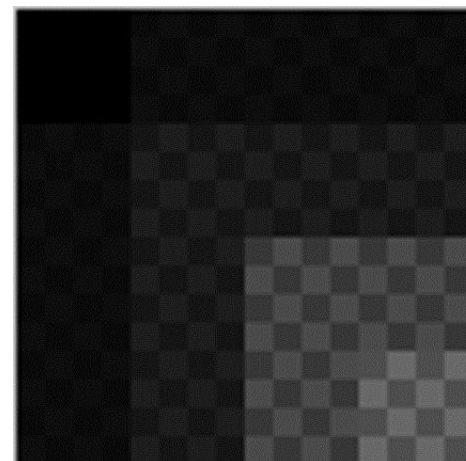


Исходный блок

255	240	240	240
240	220	220	220
240	220	180	180
240	220	180	150

Альфа исходного изображения

A ₀ = 255
A ₁ = 150
A ₂ = 240
A ₃ = 225
A ₄ = 210
A ₅ = 195
A ₆ = 180
A ₇ = 165



Восстановленное изображение

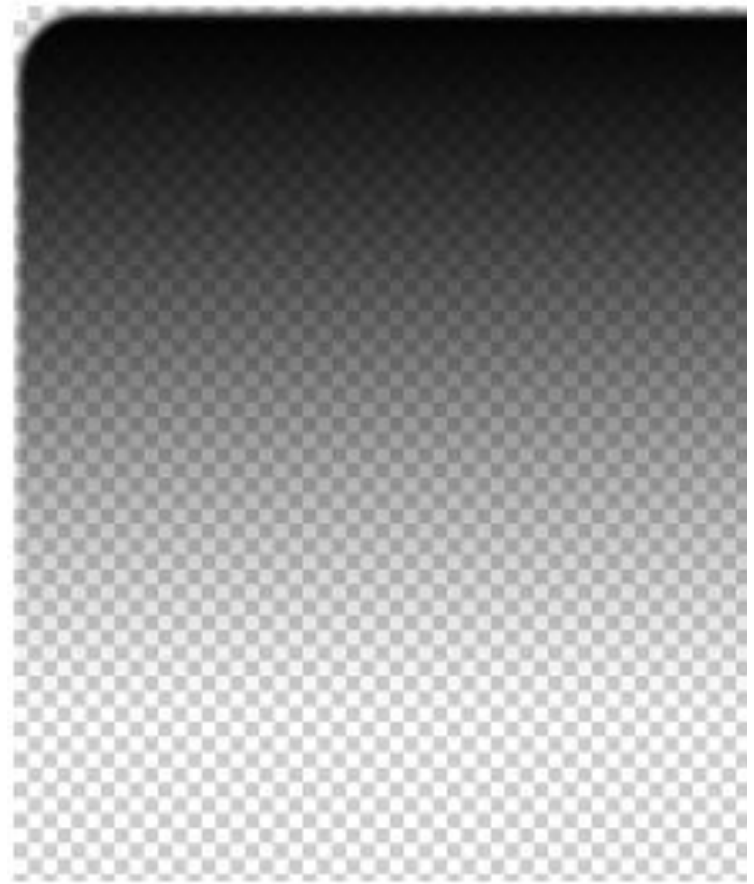
255	240	240	240
240	225	225	225
240	225	180	180
240	225	180	150

Альфа восстановленного изображения в блоке

DXT5



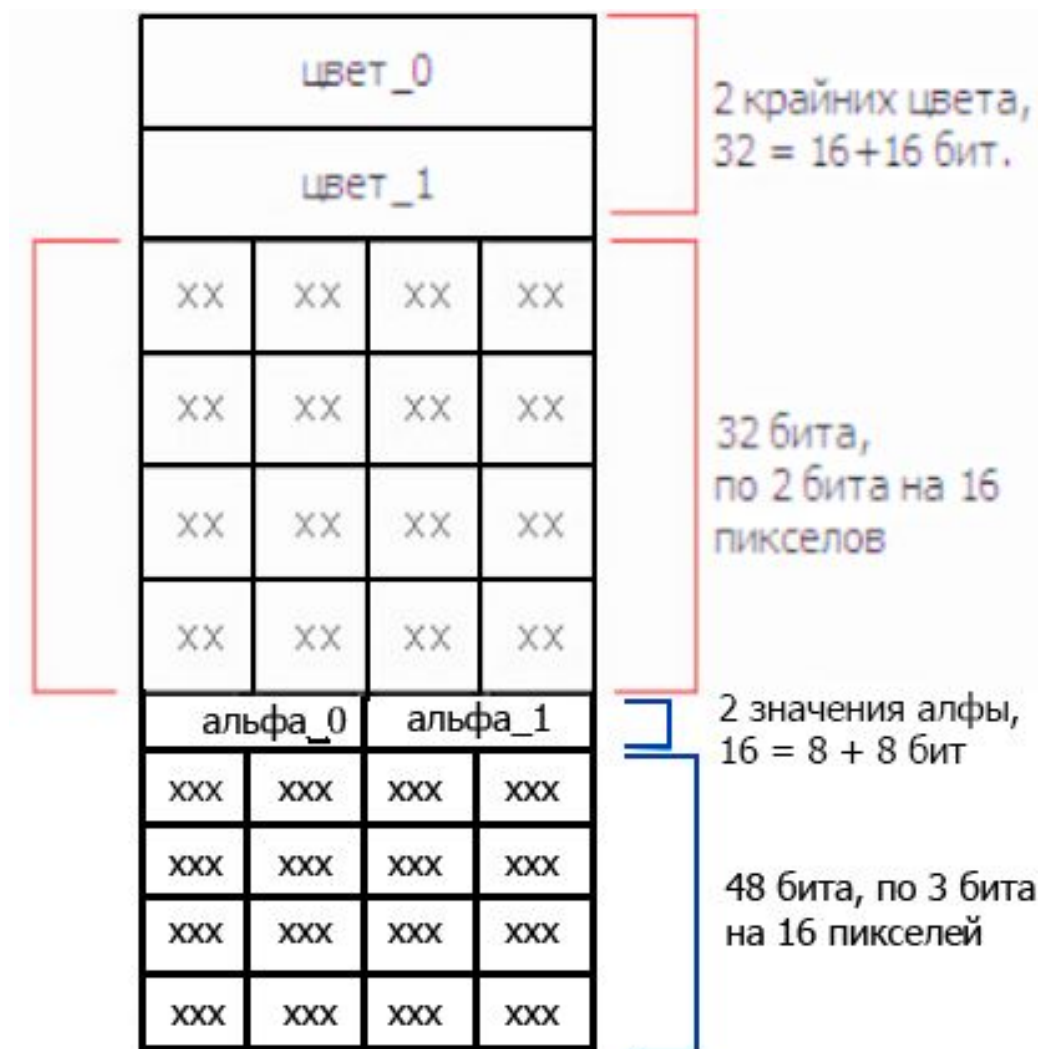
Несжатое изображение
1,47 MB



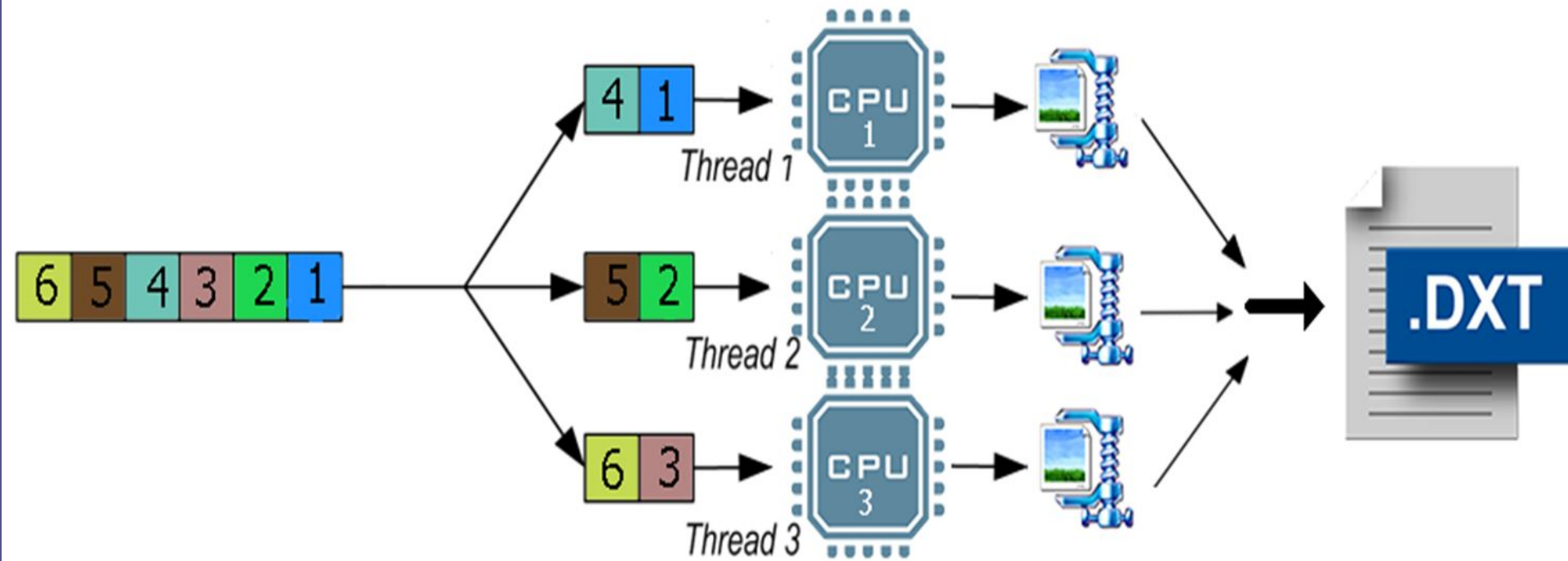
Сжатое изображение
0,35 MB



Структура DXT5 блока



Параллельный алгоритм



МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

$$\begin{cases} T = c + f \cdot x + \frac{d}{x} \rightarrow \min & (1) \\ 1 \leq x \leq P \end{cases}$$

Время работы алгоритма можно записать следующим образом:

$$d = T(m, n) = k_1 (m * n) + k_2 \quad (2)$$

c – среднее время на организацию вычислений;

f – среднее время на организацию одного потока;

x – количество потоков, задействованных в параллельном сжатии;

d - время сжатия изображения одним потоком;

P - максимальное количество активных потоков, поддерживаемых CPU;

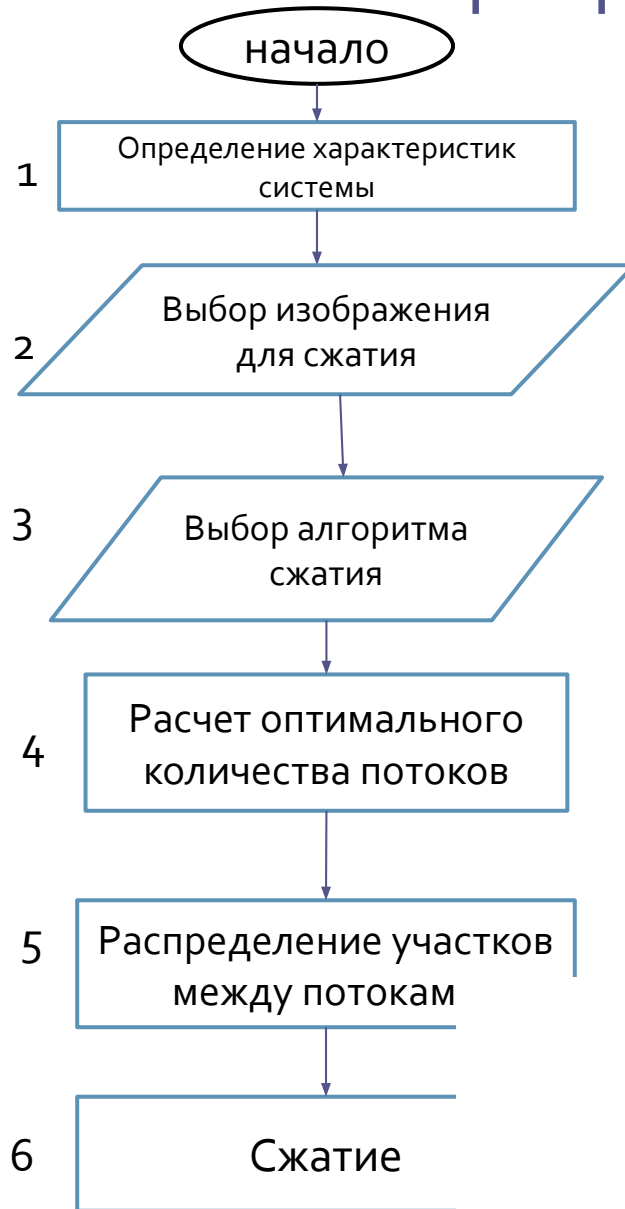
n - высота изображения;

m – ширина изображения;

k_1 – время обработки одного блока;

k_2 – среднее время на организацию вычислений

Программный комп



DXTn

Открыть Сохранить Выход

Кол-во потоков 4 Число процессоров = 4



Пример расчета оптимального количества потоков.

Дано изображение 1900x1900px. P= 4.

$$d = 1,84E-06 (n*m) + 3,42E-04.$$

c = 6,361E-03 секунд.

f - 3,459E-01 секунд.

$$T'(x) = 0 + f + d * \left(-\frac{1}{x^2}\right) = 0;$$

$$x = \sqrt{\frac{d}{f}} = \sqrt{\frac{1,84E-06(1900 * 1900) + 3,42E-04}{3,459E-01}} = 4.34$$

Т.к $x > P$, то оптимальным числом потоков будет 4. T(x) = 3.04 с.

```
for (int i = 0; i < w; i++)  
{  
    thr[i] = new Thread(new ParameterizedThreadStart(  
        thr[i].Priority = ThreadPriority.Highest;  
        thr[i].Start((object)i);  
    });  
}
```

Алгоритм

DXT 1

2

3

4

5

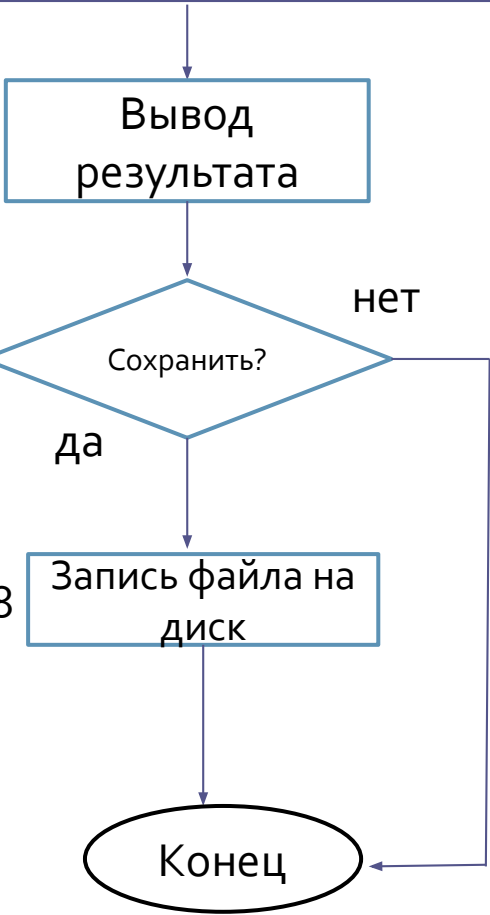
рик

картина

Б



7

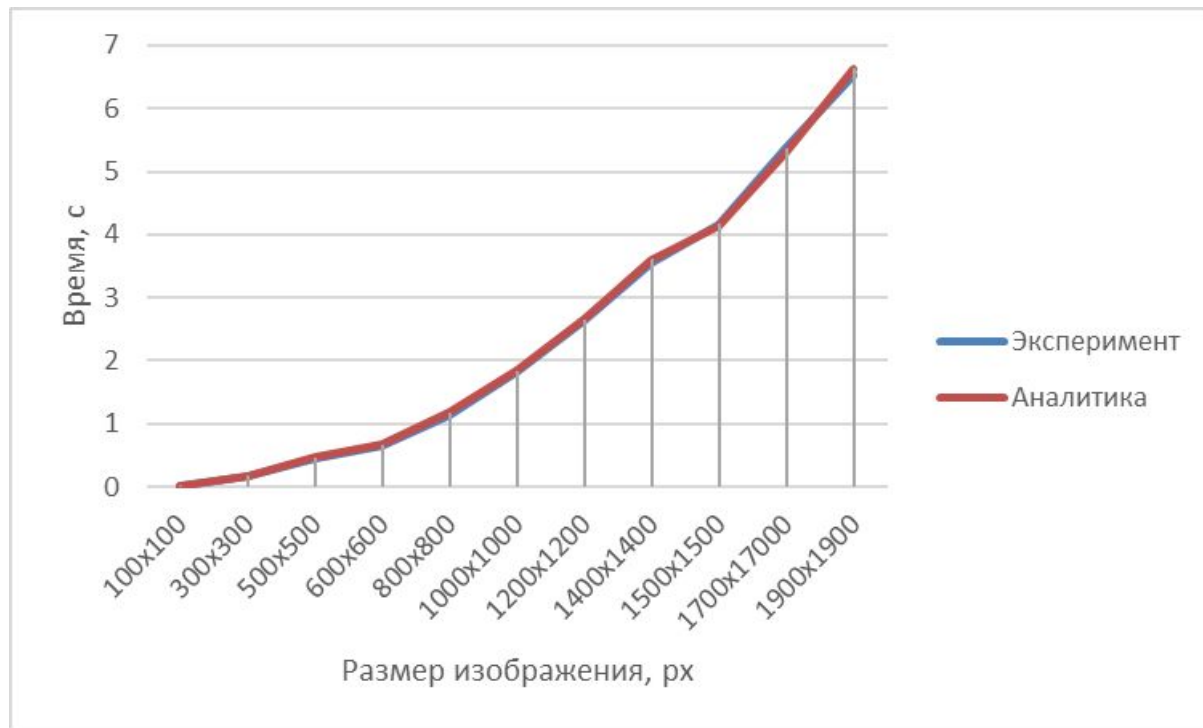


8



Эксперимент 1

Размер изображения, px	Эксперимент, с	Аналитика, с
100x100	0,02087974	0,018733
300x300	0,16296232	0,165857
500x500	0,44846666	0,460105
600x600	0,64442029	0,662401
800x800	1,1347506	1,177335
1000x1000	1,80897122	1,839394
1200x1200	2,63456706	2,648576
1400x1400	3,55396282	3,604883
1500x1500	4,16899386	4,138208
1700x1700	5,37627342	5,315201
1900x1900	6,53036216	6,639319



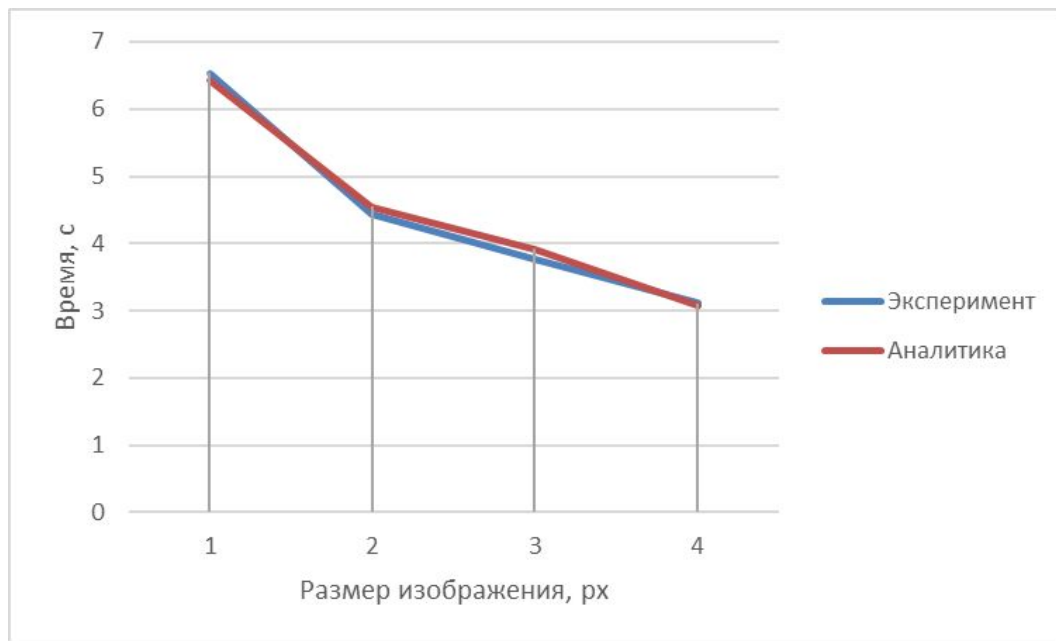
$$k_1 = 1,84E-06$$

$$k_2 = 3,42E-04$$

Эксперимент 2

DXT1

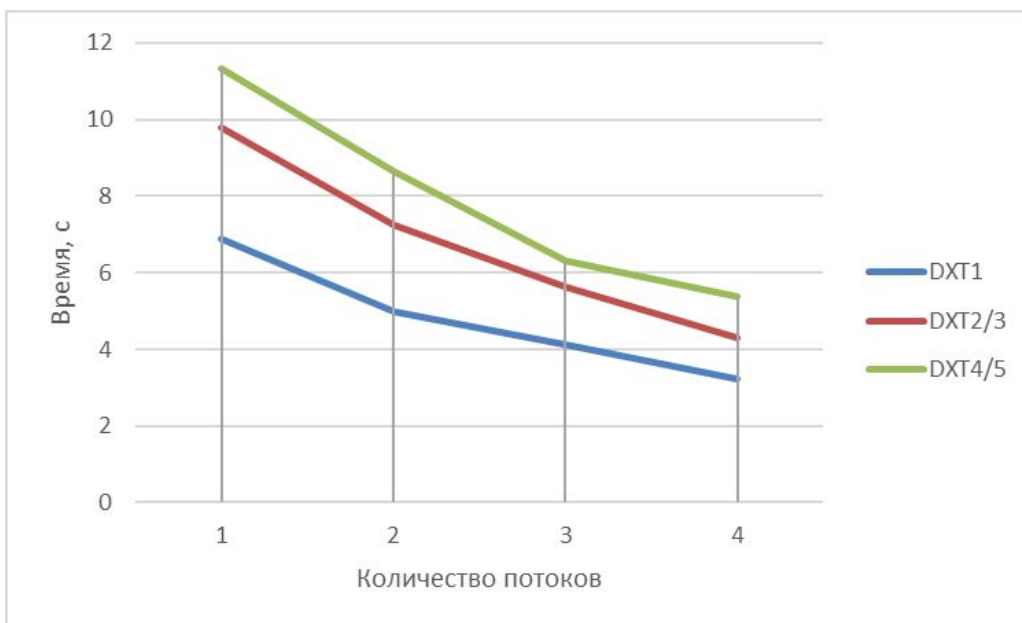
P	Эксперимент	Аналитика
1	6,53036216	6,639319
2	4,43177562	4,5483575
3	3,86364146	3,9181778
4	3,11221826	3,0392697



Эксперимент 3

Размер\поток	1	4
100x100	0,02088	0,008942
300x300	0,162962	0,079999
500x500	0,448467	0,219866
600x600	0,64442	0,313288
800x800	1,134751	0,57777
1000x1000	1,80897	0,881983
1200x1200	2,634567	1,284812
1400x1400	3,553962	1,753722
1500x1500	4,168994	1,968062
1700x1700	5,376273	2,473851
1900x1900	6,530362	3,112218





Алгоритм\потоки	1	2	3	4
DXT ₁	6,90211	4,99452	4,1231	3,22342
DXT _{2/3}	9,78657	7,23214	5,64125	4,288987
DXT _{4/5}	11,32468	8,659836	6,299455	5,383394

Заключение

- 1. Был проведен сравнительный анализ предметной области.
- 2. Была выбрана математическая модель, минимизирующая время работы параллельного алгоритма.
- 3. Был разработан параллельный алгоритм сжатия
- 4. Создан программный комплекс, реализующий параллельный DXTn алгоритмы.
- 5. Был проведен сравнительный анализ параллельных DXTn алгоритмов.