МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**Отчет по лабораторной работе № 1**

по дисциплине: ”Распределенная и параллельная обработка данных”

на тему: ***”*** ***Разработка программы моделирования закона Амдала ”***

Выполнил**:** студент группы 10701214 Зубарев А. А.

Принял**:** проф. Прихожий А.А.

Минск 2017

# Лабораторная работа 1. Разработка программы моделирования закона Амдала.

**Цель работы:** Изучить основной и сетевой закон Амдала.

## Задание

Спроектировать программу для построения графиков зависимостей ускорения от a, n, Ca, Ct.

**Теоретическая часть**

Закон Амдала устанавливает величину ускорения вычислений на многопроцессорной системе (N) по сравнению с однопроцессорной системой с зависимостью от решаемой задачи и алгоритма и, с другой стороны, от объема представленных вычислительных ресурсов.

**Базовый закон Амдала**

Описывается формулой:

Где R – ускорение вычислений;

Wпосл – те операции, которые выполняются взаимопоследовательно;

Wпар – количество операторов, которые могут выполняться параллельно;

W – общее количество операций;

n – количество процессоров.

**Сетевой закон Амдала**

Описывается формулой:

Где Rс – ускорение вычислений;

Wс – операции обмена данными;

tс – среднее время обмен данными;

с – коэффициент деградации;

Са – деградация по количеству операций;

Сt – деградация по времени.

### Код программы

using System;

using System.Windows.Forms;

namespace Lab\_1\_\_RiPOD

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private double check\_double(TextBox TB)

{

double res = 0;

string text = TB.Text;

try

{

res = double.Parse(text);

}

catch

{

try

{

string s = text.Replace('.', ',');

res = double.Parse(s);

TB.Text = s;

}

catch

{

TB.Text = "Error";

res = 0;

}

}

return res;

}

private void button\_ZA\_R\_a\_Click(object sender, EventArgs e)

{

double a\_1 = 0;

double a\_2 = 0;

int n;

double h;

double R = 0;

h = check\_double(textBox\_step\_1\_1);

a\_1 = check\_double(textBox\_a\_1\_1\_1);

a\_2 = check\_double(textBox\_a\_1\_1\_2);

try

{

n = int.Parse(textBox\_n\_1\_1.Text);

}

catch

{

textBox\_n\_1\_1.Text = "Error";

return;

}

richTextBox1.Text = "";

chart1.Series[0].Points.Clear();

while ((a\_1 <= a\_2) && h > 0)

{

R = 1 / (a\_1 + (1 - a\_1) / n);

richTextBox1.Text += " a = " + a\_1 + " R = " + R + "\n";

chart1.Series[0].Points.AddXY(a\_1, R);

a\_1 += h;

a\_1 = Math.Round(a\_1, 5);

}

}

private void button\_ZA\_R\_n\_Click(object sender, EventArgs e)

{

int n\_1 = 0;

int n\_2 = 0;

double a;

int h;

double R = 0;

try

{

h = int.Parse(textBox\_step\_1\_2.Text);

n\_1 = int.Parse(textBox\_n\_1\_2\_1.Text);

n\_2 = int.Parse(textBox\_n\_1\_2\_2.Text);

}

catch

{

textBox\_step\_1\_2.Text = "Error";

textBox\_n\_1\_2\_1.Text = "Error";

textBox\_n\_1\_2\_2.Text = "Error";

textBox\_a\_1\_2.Text = "Error";

return;

}

a = check\_double(textBox\_a\_1\_2);

richTextBox2.Text = "";

chart2.Series[0].Points.Clear();

while ((n\_1 <= n\_2) && h > 0)

{

R = 1 / (a + (1 - a) / n\_1);

richTextBox2.Text += " n = " + n\_1 + " R = " + R + "\n";

chart2.Series[0].Points.AddXY(n\_1, R);

n\_1 += h;

}

}

private void button\_ZA\_Rc\_Ca\_Click(object sender, EventArgs e)

{

int n;

double Ca\_1 = 0;

double Ca\_2 = 0;

double Ct;

double a;

double h;

double c;

double Rc = 0;

try

{

n = int.Parse(textBox\_n\_2\_1.Text);

}

catch

{

textBox\_n\_2\_1.Text = "Error";

return;

}

Ca\_1 = check\_double(textBox\_Ca\_2\_1\_1);

Ca\_2 = check\_double(textBox\_Ca\_2\_1\_2);

Ct = check\_double(textBox\_Ct\_2\_1);

a = check\_double(textBox\_a\_2\_1);

h = check\_double(textBox\_step\_2\_1);

richTextBox3.Text = "";

chart3.Series[0].Points.Clear();

while ((Ca\_1 <= Ca\_2) & h > 0)

{

c = Ct \* Ca\_1;

Rc = 1 / (a + (1 - a) / n + c);

richTextBox3.Text += " Ca = " + Ca\_1 + " Rc = " + Rc + "\n";

chart3.Series[0].Points.AddXY(Ca\_1, Rc);

Ca\_1 += h;

Ca\_1 = Math.Round(Ca\_1, 5);

}

}

private void button\_ZA\_Rc\_Ct\_Click(object sender, EventArgs e)

{

int n;

double Ct\_1 = 0;

double Ct\_2 = 0;

double Ca;

double a;

double h;

double c;

double Rc = 0;

try

{

n = int.Parse(textBox\_n\_2\_2.Text);

}

catch

{

textBox\_n\_2\_2.Text = "Error";

return;

}

Ct\_1 = check\_double(textBox\_Ct\_2\_2\_1);

Ct\_2 = check\_double(textBox\_Ct\_2\_2\_2);

Ca = check\_double(textBox\_Ca\_2\_2);

a = check\_double(textBox\_a\_2\_2);

h = check\_double(textBox\_step\_2\_2);

richTextBox4.Text = "";

chart4.Series[0].Points.Clear();

while ((Ct\_1 <= Ct\_2) & h > 0)

{

c = Ct\_1 \* Ca;

Rc = 1 / (a + (1 - a) / n + c);

richTextBox4.Text += " Ct = " + Ct\_1 + " Rc = " + Rc + "\n";

chart4.Series[0].Points.AddXY(Ct\_1, Rc);

Ct\_1 += h;

Ct\_1 = Math.Round(Ct\_1, 5);

}

}

}

}

### Скриншоты результатов

Рисунок 1 – базовый закон Амдала с зависимостью R от a

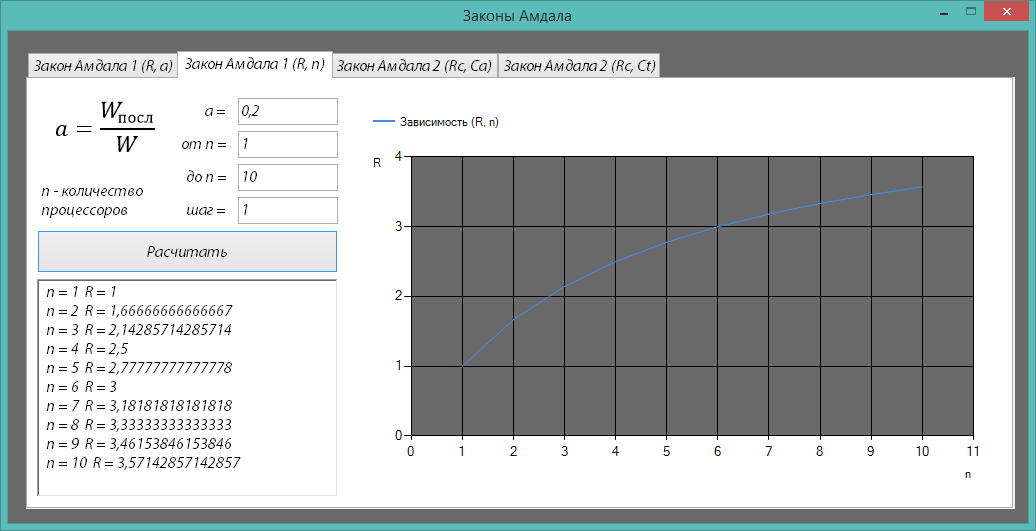
Из анализа рисунка 1 следует, что чем больше количество операций, которые выполняются взаимопоследовательно, тем меньше ускорение.

Рисунок 2 – базовый закон Амдала с зависимостью R от n

Из анализа рисунка 2 следует, что чем больше количество процессоров, тем больше ускорение.

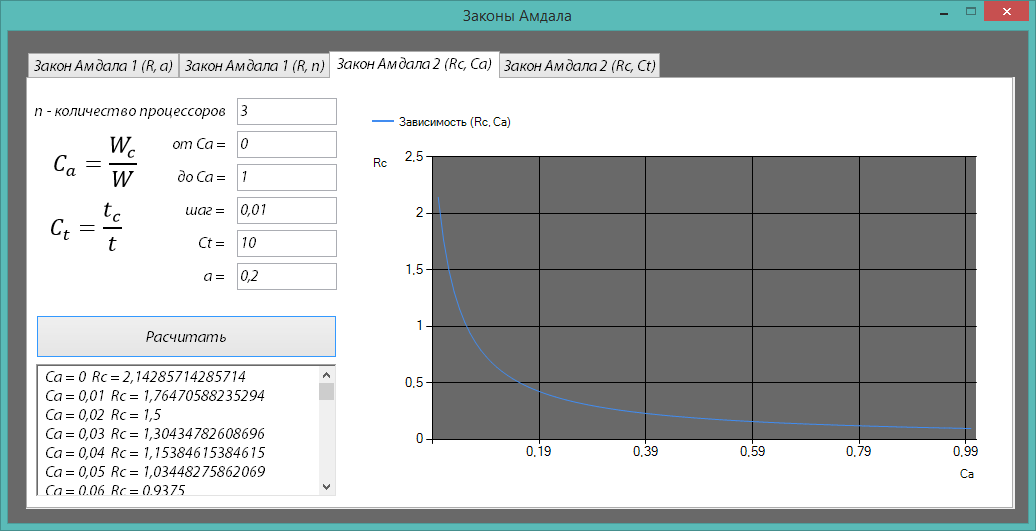


Рисунок 3 – сетевой закон Амдала с зависимостью R от Са

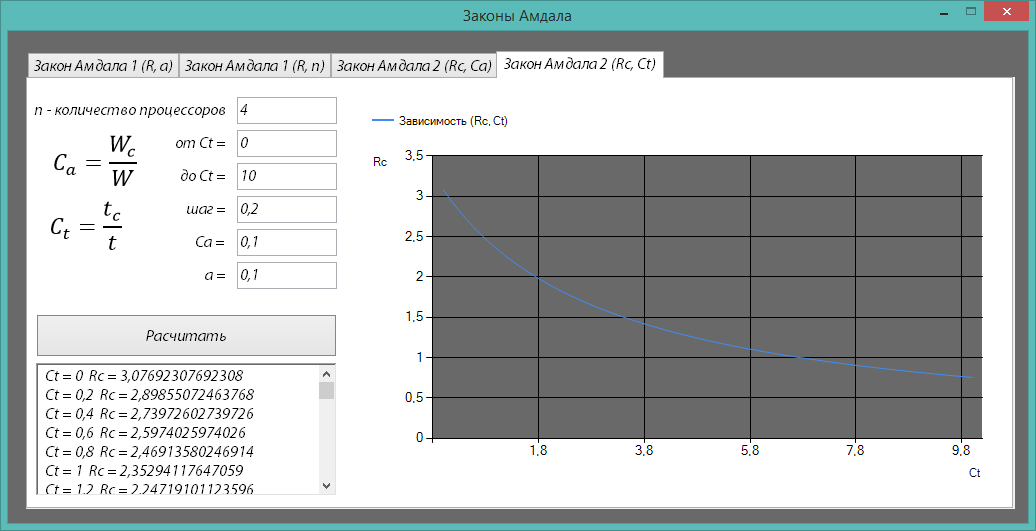
Из анализа рисунка 3 следует, что чем больше количество операций сетевого обмена данными, тем меньше ускорение.

Рисунок 4 – сетевой закон Амдала с зависимостью R от Ct

Из анализа рисунка 4 следует, что чем больше среднее время сетевого обмена данными, тем меньше ускорение.

## Вывод

Изучены обычный и сетевой законы Амдала.