ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет по лабораторной работе №2**

**по дисциплине: «Методы распараллеливания программ»**

**Тема: «Реализация алгоритма для систем с общей памятью»**

студента очного отделения

3 курса 12001801 группы

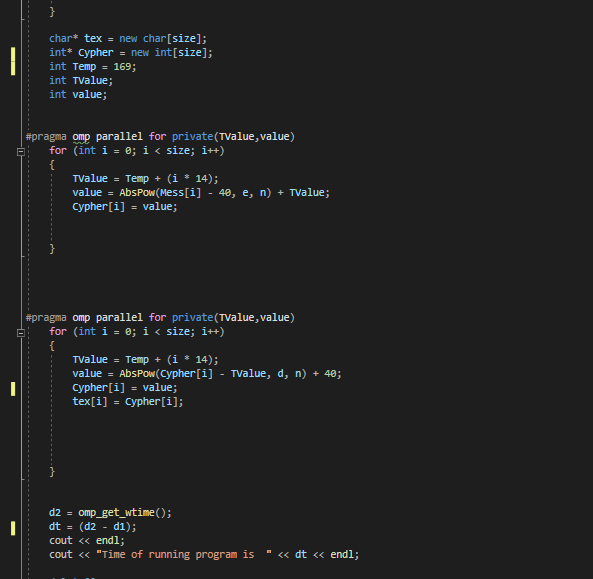
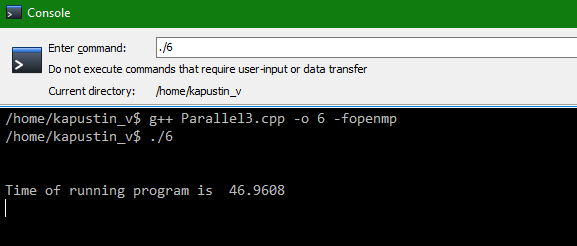
Капустина Виктора Сергеевича

Проверил(а):

Петров Денис Васильевич

Белгород 2020

## **16 Вариант Блок схема**

  
Рис. 1. Процесс разработки программного обеспечения  
  
Рис. 2. Работа приложения

## Таблица 1. Зависимость времени вычислений от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Время расчета на 1 ядре, сек. | Время расчета на 2 ядрах, сек. | Время расчета на 3 ядрах, сек. | Время расчета на 4 ядрах, сек. | Время расчета на 5 ядрах, сек. | Время расчета на 6 ядрах, сек. |
| **16** | 0.0003590 | 0.0006275 | 0.0007765 | 0.0009641 | 0.0007922 | 0.0012408 |
| **1600** | 0.002128 | 0.002226 | 0.0021546 | 0.0018476 | 0.0018272 | 0.001823 |
| **160000** | 0.12296 | 0.11739 | 0.0866185 | 0.0715651 | 0.0599274 | 0.0534555 |
| **16000000** | 8.41106 | 7.90351 | 5.70245 | 4.63107 | 3.84684 | 3.47589 |
| **160000000** | 83.5575 | 83.1292 | 57.1754 | 46.9608 | 38.0264 | 32.7543 |

## Таблица 1.1 Зависимость времени вычислений от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Время расчета на 7 ядрах, сек. | Время расчета на 8 ядрах, сек. | Время расчета на 9 ядрах, сек. | Время расчета на 10 ядрах, сек. | Время расчета на 11 ядрах, сек. | Время расчета на 12 ядрах, сек. |
| **16** | 0.0014443 | 0.0014943 | 0.001384 | 0.0017777 | 0.002001 | 0.0019002 |
| **1600** | 0.0015772 | 0.0017539 | 0.0019769 | 0.0019796 | 0.0020707 | 0.0023551 |
| **160000** | 0.0487921 | 0.0439132 | 0.0404682 | 0.0409945 | 0.0356269 | 0.0335426 |
| **16000000** | 3.00767 | 2.6784 | 2.47325 | 2.30505 | 2.19712 | 2.02547 |
| **160000000** | 28.5918 | 25.909 | 24.2878 | 22.1634 | 20.0835 | 18.1363 |

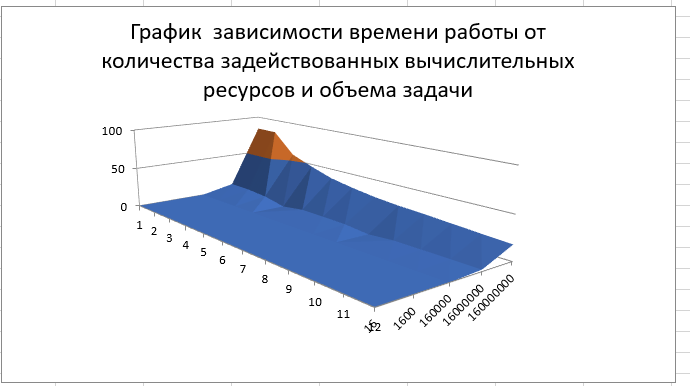
  
Рис. 3 Графика зависимости времени расчетов от количества задействованных вычислительных ресурсов и объема задачи

Таблица 2. Зависимость ускорения от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Коэффициент ускорения K на 1 ядре | Коэффициент ускорения K на 2 ядрах | Коэффициент ускорения K на 3 ядрах | Коэффициент ускорения K на 4 ядрах | Коэффициент ускорения K на 5 ядрах | Коэффициент ускорения K на 6 ядрах |
| **16** | 1 | 0.572111 | 0.462330 | 0.382368 | 0.453168 | 0.289329 |
| **1600** | 1 | 0.955974 | 0.987654 | 1.151764 | 1.164623 | 1.167306 |
| **160000** | 1 | 1.047430 | 1.149558 | 1.718155 | 2.051816 | 2.300231 |
| **16000000** | 1 | 1.064218 | 1.474990 | 1.816223 | 2.186485 | 2.419829 |
| **160000000** | 1 | 1.005152 | 1.461423 | 1.779314 | 2.197354 | 2.551039 |

Таблица 2.1 Зависимость ускорения от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Коэффициент ускорения K на 7 ядрах | Коэффициент ускорения K на 8 ядрах | Коэффициент ускорения K на 9 ядрах | Коэффициент ускорения K на 10 ядрах | Коэффициент ускорения K на 11 ядрах | Коэффициент ускорения K на 12 ядрах |
| **16** | 0.248563 | 0.240246 | 0.259393 | 0.201946 | 0.179410 | 0.188927 |
| **1600** | 1.349226 | 1.213296 | 1.076432 | 1.074964 | 1.027671 | 0.903570 |
| **160000** | 2.52008 | 2.800069 | 3.038435 | 2.999426 | 3.451324 | 3.665786 |
| **16000000** | 2.796536 | 3.14033 | 3.400812 | 3.451324 | 3.82822 | 4.152646 |
| **160000000** | 2.922428 | 3.225037 | 3.440307 | 3.770066 | 4.160504 | 4.607196 |

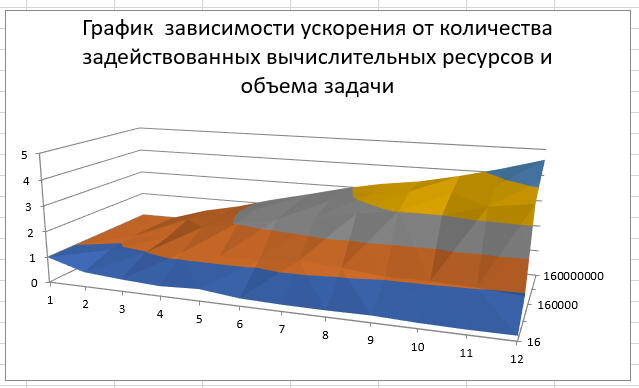


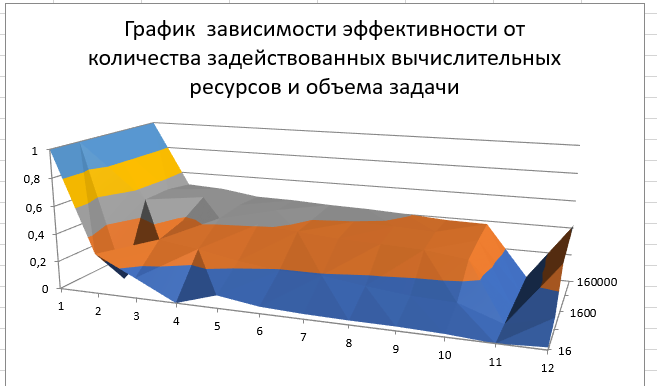
Рис. 4 График зависимости ускорения от количества задействованных вычислительных ресурсов и объема задачи

Таблица 3. Зависимость эффективности от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Коэффициент ускорения K на 1 ядре | Коэффициент ускорения K на 2 ядрах | Коэффициент ускорения K на 3 ядрах | Коэффициент ускорения K на 4 ядрах | Коэффициент ускорения K на 5 ядрах | Коэффициент ускорения K на 6 ядрах |
| **16** | 1 | 0,286055 | 0,14411 | 0.095592 | 0,090633 | 0,0482215 |
| **1600** | 1 | 0.477987 | 0,329218 | 0,287941 | 0,232924 | 0,194551 |
| **160000** | 1 | 0,523715 | 0,383186 | 0,429538 | 0,410363 | 0,383371 |
| **16000000** | 1 | 0,532109 | 0,491166 | 0,454055 | 0,437297 | 0,403304 |
| **160000000** | 1 | 0,502576 | 0,487141 | 0,444828 | 0,439470 | 0,425173 |

Таблица 3.1 Зависимость эффективности от объема исходных данных и количества задействованных вычислительных ядрах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Объем задачи** | Коэффициент ускорения K на 7 ядрах | Коэффициент ускорения K на 8 ядрах | Коэффициент ускорения K на 9 ядрах | Коэффициент ускорения K на 10 ядрах | Коэффициент ускорения K на 11 ядрах | Коэффициент ускорения K на 12 ядрах |
| **16** | 0,03550 | 0,03 | 0,02882 | 0,020194 | 0.179410 | 0,01717 |
| **1600** | 0,19274 | 0,151662 | 0,1196 | 0,107496 | 1.027671 | 0,082142 |
| **160000** | 0,36001 | 0,35 | 0,3376 | 0,299942 | 3.451324 | 0,333253 |
| **16000000** | 0,39950 | 0,392541 | 0,37786 | 0,345132 | 3.82822 | 0,377513 |
| **160000000** | 0,41748 | 0,40312 | 0,38225 | 0,377006 | 4.160504 | 0,418836 |

  
Рис. 5 График зависимости эффективности от количества задействованных вычислительных ресурсов и объема задачи

**Листинг программы**

#include <iostream>

#include <cstdlib>

#include <cctype>

#include <string>

#include <bitset>

#include <cmath>

#include <time.h>

#include <ctime>

#include <string>

#include <iomanip>

#include <omp.h>

using namespace std;

int gcd(int x, int y)

{

return y ? gcd(y, x % y) : x;

}

int To\_Double\_state(int num) // перевод в двоичную степень

{

int t = 0, g = 1;

while (num)

{

t += (num % 2) \* g;

num = num / 2;

g = g \* 10;

}

return t;

}

int Len\_Number(int num) // расчет длины числа в двоичной степени

{

int t = 1;

while ((num / 10) != 0)

{

t++;

num = num / 10;

}

return t;

}

int AbsPow(int number, int pw, int rem) //Алгоритм быстрого возведения в степень по модулю

{

int Doub = To\_Double\_state(pw);

int quant = Len\_Number(Doub);

int\* arr = new int[quant];

arr[0] = number;

for (int i = 1; i < quant; i++)

{

arr[i] = (arr[i - 1] \* arr[i - 1]) % rem;

}

number = 1;

for (int i = 0; i < quant; i++)

{

number = number \* (pow((arr[i]), Doub % 10));

Doub = Doub / 10;

}

number = number % rem;

delete[] arr;

return number;

}

int main()

{

double d1, d2, dt;

d1 = omp\_get\_wtime();

omp\_set\_num\_threads(4); //Кол-во нитей

string str;

srand(time(0));

setlocale(LC\_ALL, "Russian");

int p, q, n, fi;

int e = 0;

int rnd = rand() % 2;

int size = 2000000; //Размер сообщения

char\* Mess = new char[size];

for (int i = 0; i < size; i++)

{

Mess[i] = 65 + rand() % 57;

}

cout << endl;

switch (rnd)

{

case 0: p = 7; break;

case 1: p = 13; break;

}

q = 17;

n = abs(p \* q);

fi = (p - 1) \* (q - 1);

int i = 2;

while (e < 1) {

if ((gcd(i, fi) == 1) && (i != p) && (i != q))

{

e = i;

}

i++;

}

bool flag = false;

int ff = 0;

int d = 1;

while (flag != true)

{

if (((d \* e) % fi) == 1)

{

if (ff == 1)

{

flag = true;

}

ff = 1;

}

else d++;

}

char\* tex = new char[size];

int\* Cypher = new int[size];

int Temp = 169;

int TValue;

int value;

#pragma omp parallel for private(TValue,value)

for (int i = 0; i < size; i++)

{

TValue = Temp + (i \* 14);

value = AbsPow(Mess[i] - 40, e, n) + TValue;

Cypher[i] = value;

}

#pragma omp parallel for private(TValue,value)

for (int i = 0; i < size; i++)

{

TValue = Temp + (i \* 14);

value = AbsPow(Cypher[i] - TValue, d, n) + 40;

Cypher[i] = value;

tex[i] = Cypher[i];

}

d2 = omp\_get\_wtime();

dt = (d2 - d1);

cout << endl;

cout << "Time of running program is " << dt << endl;

delete[] Mess;

delete[] Cypher;

delete[] tex;

system("pause");

return 0;

}