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 5**

**по дисциплине: «Модели параллельного программирования»**

**Тема работы «Знакомство с технологией MPI»**

студента очного отделения

2 курса 12001801 группы

Капустина Виктора Сергеевича

Проверил(а):

Петров Денис Васильевич

Белгород 2020

**Цель:** Получить практический навык использования технологии MPI при решении прикладных задач.

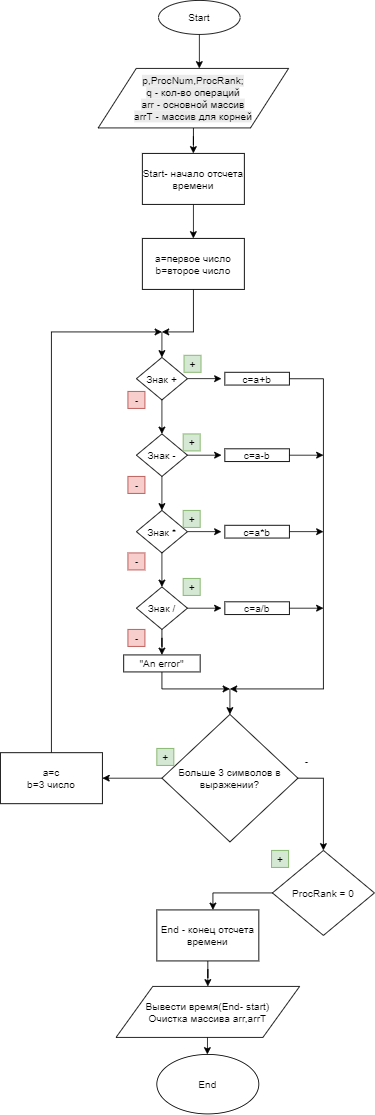
**Индивидуальное задание. 17 вариант.** Дана последовательность арифметических выражений, операндами которых являются однозначные числа, а число операций не больше двух. Найти значения всех выражений.

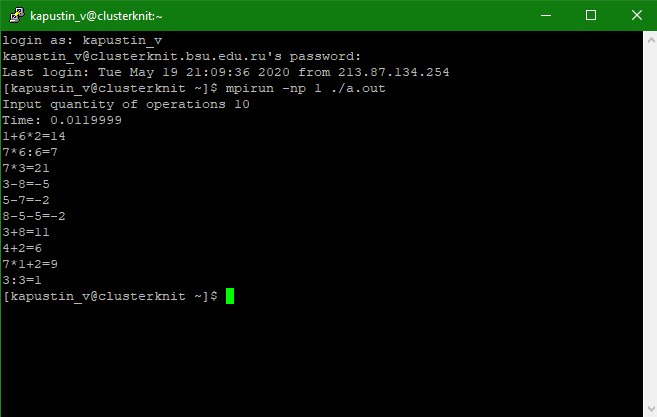
**Теоретическая часть**

MPI технология в программе начинается с инициализации параллельной области с помощью функции *MPI\_Init*. Далее с помощью функции *MPI\_Comm\_size* определяется количество процессов, а с помощью функции *MPI\_Comm\_rank* определяется ранг процесса.

Передача данных между узлами возможна в основном через коллективные операции передачи. Обычно с корневого узла данные передаются на другие узлы*(Bcast,Scatter)*, где и происходят расчеты, а потом результаты собираются обратно в корневой узел*(Reduce,Gather)*.

**Блок схема**

****

**  
Скриншот работы программы в тестовом режиме на 1 узле**

**Листинг 1**

#include<iostream>

#include<ctime>

#include<stdlib.h>

#include <mpi.h>//подключение библиотеки mpi

using namespace std;

int main(int agrc, char\* argv[])

{

setlocale(LC\_ALL, "Rus");

int ProcNum, ProcRank;

double start, end;

int q = 17;

int p;

int\*\* arr = new int\* [q];

for (int i = 0; i < q; i++) {

arr[i] = new int[7];

}

int\*\* arrT = new int\* [q];

for (int i = 0; i < q; i++) {

arrT[i] = new int[7];

}

MPI\_Init(&agrc, &argv);//инициализация начала MPI-программы

MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);//функция расчёта количества процессоров

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);//функция расчёта ранга процессоров

MPI\_Scatter(&arr, 1, MPI\_INT, &arrT, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

start = MPI\_Wtime();

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

{

p = 2 + rand() % 2;

arr[i][0] = p \* 2 - 1;

for (int j = 1; j <= arr[i][0]; j++)

{

switch (j % 2)

{

case 1:

arr[i][j] = 1 + rand() % 8;

break;

case 0:

arr[i][j] = 1 + rand() % 4;

break;

default:

cout << "Не выполнены условия";

break;

}

}

int a = arr[i][1];

int b = arr[i][3];

int c;

switch (arr[i][2])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

if (arr[i][0] > 3)

{

a = c;

b = arr[i][5];

switch (arr[i][4])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

}

arr[i][6] = c;

}

MPI\_Gather(&arrT, 1, MPI\_INT, &arr, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

// MPI\_Reduce(&arr, &arr, 1, MPI\_INT, MPI\_MAX, 0, MPI\_COMM\_WORLD);//отправка сообщения от всех процессов программы одному процессу программы

if (ProcRank == 0)

{

end = MPI\_Wtime();//конец отсчёта времени

cout << "Time: " << (end - start) \* 1000 << endl;

for (int i = 0; i < q; i++) {

delete[] arr[i];

}

delete[] arr;

for (int i = 0; i < q; i++) {

delete[] arrT[i];

}

delete[] arrT;

}

MPI\_Finalize();

}

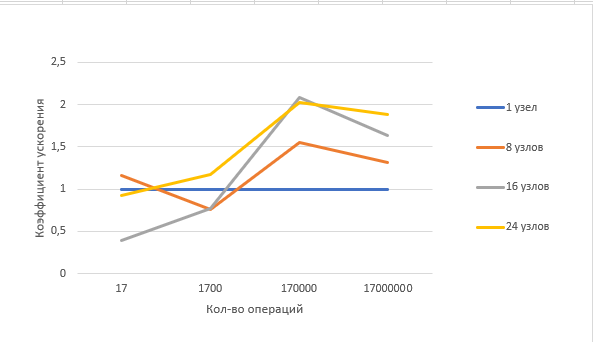
**Конец листинга 1**

**Таблица №1. Время расчета работы программы**

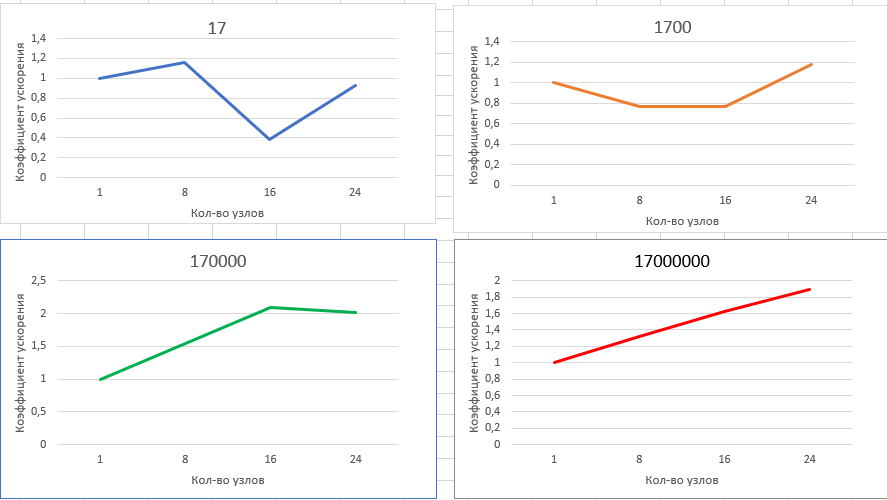
| Номер расчета | Количество  итераций | Время расчета на 1 узле | Время расчета на 8 узлах | Время расчета на 16 узлах | Время расчета на 24 узлах |
| --- | --- | --- | --- | --- | --- |
| 1 | 17 | 0.0229999 | 0.198 | 0.5924 | 0.247 |
| 2 | 1700 | 0.538 | 0.7045 | 0.698 | 0.457 |
| 3 | 170000 | 44.650 | 28.784 | 21.4 | 22.058 |
| 4 | 17000000 | 2914.588 | 2214.27 | 1789.44 | 1543.04 |

**Таблица №2. Коэффициент ускорения ( Кi= Ti1/ Ti j, где i – номер расчета, j - кол-во узлов)**

| № | P – количество расчетов | Коэффициент ускорения К на 1 узле | Коэффициент ускорения К на 8 узлах | Коэффициент ускорения К на 16 узлах | Коэффициент ускорения К на 24 узлахх |
| --- | --- | --- | --- | --- | --- |
| 1 | 17 | 1 | 1,1611 | 0,3880 | 0,9307 |
| 2 | 1700 | 1 | 0,7636 | 0,770 | 1,177 |
| 3 | 170000 | 1 | 1,5512 | 2,0864 | 2,0242 |
| 4 | 17000000 | 1 | 1,3162 | 1,6287 | 1,888 |

**График 1. График зависимости коэффициента ускорения от количества операций.  
**

**График 1. График зависимости коэффициента ускорения от количества узлов.**



**Вывод:** Ознакомился с технологией MPI и получил практический навык использования технологии MPI при решении прикладных задач. При маленьких значениях много времени уходит на создание потоков, поэтому скорость работы увеличивается в основном при увеличении количества итераций.