**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Параллельное программирование

**Отчет по лабораторной работе № 1**

**Тема:** «Параллельное вычисление суммы числового ряда»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМИ-354 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Мова И.А |  |  |  |
| Принял | Спеле В.В. |  |  |  |

**Уфа 2023**

**Цель:** для многопроцессорных вычислительных систем с распределенной памятью на примере задачи параллельного вычисления суммы числового ряда научиться программно реализовывать простейшие параллельные вычислительные алгоритмы и проводить анализ их эффективности.

**Теоретический материал.**

*MPI (Message Passing Interface****)*** – это стандартизованная библиотека функций, призванная обеспечить совместную работу параллельных процессов путем организации передачи сообщений между ними.

*Основные понятия MPI*

* Каждая программа представляет собой совокупность одновременно работающих процессов, которые могут обмениваться сообщениями;
* Все процессы объединяются в группы;
* Обмен сообщениями возможен между процессами одной группы, которой поставлена в соответствие своя область связи;
* Коммуникатор – идентификатор области;
* При запуске программы все доступные ей процессы объединяются в начальную группу с общей областью связи, имеющей коммуникатор MPI\_COMM\_WORLD.

*Особенности реализации MPI для C/C++*

1. Первая строка программы: #include “mpi.h”;
2. В MPI принят ANSI C стандарт;
3. Нумерация массивов начинается с 0;
4. Массивы хранятся по строкам;
5. Логические переменные являются переменными типа integer со значением 0 в случае false и любым не нулевым значением, обозначающем true.

*Базовые функции MPI*

1. int MPI\_Init(int \*argc, char \*\*argv[]); – инициализация параллельной части программы.

Возвращает предопределенные константы

MPI\_SUCCESS – возвращается в случае успешного выполнения,

MPI\_ERR\_ARG – ошибка неправильного задания аргумента,

MPI\_ERR\_INTERN – внутренняя ошибка (нехватка памяти),

MPI\_ERR\_UNKNOWN – неизвестная ошибка.

1. int MPI\_Finalize(void); – завершение параллельной части программы.
2. int MPI\_Comm\_size(MPI\_Comm comm, int\* size); – определение числа процессов size в коммуникационной группе с коммуникатором comm.
3. int MPI\_Comm\_rank(MPI\_comm comm, int\* rank); – определение номера rank вызвавшего ее процесса, входящего в коммуникационную группу с коммуникатором comm
4. int MPI\_Send(void\* sbuf, int count, MPI\_Datatype datatype, int dest, int tag, MPI\_Comm comm) – передача сообщений от одного процесса к другому

sbuf – адрес в памяти, начиная с которого размещаются передаваемые данные;

count – количество передаваемых элементов;

datatype – тип передаваемых элементов;

dest – номер процесса-получателя сообщения;

tag – метка передаваемого сообщения;

comm – коммуникатор.

1. int MPI\_Recv(void\* rbuf, int count, MPI\_Datatype datatype, int source, int tag, MPI\_comm comm, MPI\_Status \*status) – прием сообщений от одного процесса к другому.

*Входные параметры:*

count – количество получаемых элементов;

datatype – тип получаемых элементов;

source – номер процесса-отправителя сообщения;

tag – метка принимаемого сообщения;

comm – коммуникатор.

*Выходные параметры:*

rbuf – адрес в памяти, начиная с которого размещаются принимаемые данные;

status – структура, содержащая информацию о принятом сообщении.

Структура status имеет три поля.

Status.MPI\_SOURCE – номер процесса-отправителя;

Status.MPI\_TAG – метка принимаемого сообщения;

Status.MPI\_ERROR – код завершения приема сообщения.

*Функция совмещенного приема/передачи.*

int MPI\_Sendrecv(void\* sbuf, int scount, MPI\_Datatype sdatatype, int dest, int stag, void\* rbuf, int rcount, MPI\_Datatype rdatatype, int source, int rtag, MPI\_comm comm, MPI\_Status \*status);

*Входные параметры:*

sbuf – адрес в памяти, начиная с которого размещаются передаваемые данные;

scount – количество передаваемых элементов;

sdatatype – тип отправляемых элементов;

dest – номер процесса-получателя сообщения;

stag – метка отправляемого сообщения;

rcount – количество получаемых элементов;

rdatatype – тип получаемых элементов;

source – номер процесса-отправителя сообщения;

rtag – метка принимаемого сообщения;

comm – коммуникатор.

*Выходные параметры:*

rbuf – адрес в памяти, начиная с которого размещаются принимаемые данные;

status – структура, содержащая информацию о принятом сообщении.

Для анализа данных, полученных в ходе проведения лабораторной работы, будут использоваться следующие определения:

* Отношение времени выполнения параллельной программы на одном процессоре (ядре) ко времени выполнения параллельной программы на процессорах называется ускорением при использовании p процессоров:

* Отношение ускорения к количеству процессоров называется эффективностью при использовании процессоров:

**Практическая часть**

Была написана программа, которая вычисляет сумму следующего ряда

Были подобраны такие , при которых время работы программы составило 150 с ( ) и 300 с ().

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 150, 012757 | 261,008087 |
| 2 | 80,008284 | 133,351072 |
| 4 | 40,018469 | 66,686364 |
| 8 | 20,021974 | 33,370369 |
| 16 | 10,012725 | 16,685254 |
| 32 | 5,005005 | 8,345877 |
| 64 | 2,503040 | 4,171902 |

*Таблица 1. Результат замеров времени при разном числе процессов*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
|  |  |  |  |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1,8749653 | 2,249761 | 0,937483 | 0,97865 |
| 4 | 3,7485881 | 4,498792 | 0,937147 | 0,978491 |
| 8 | 7,4924059 | 8,990254 | 0,936551 | 0,977694 |
| 16 | 14,982211 | 17,98043 | 0,936388 | 0,97769 |
| 32 | 29,972549 | 35,94686 | 0,936642 | 0,977309 |
| 64 | 59,932225 | 71,91159 | 0,936441 | 0,977552 |

*Таблица 2. Ускорение и эффективность при разном числе процессов*

*Рисунок 1. Ускорение при разном числе процессов*

*Рисунок 2. Эффективность при разном числе процессов (при и )*

**Вывод**: в ходе лабораторной работы был реализован алгоритм вычисления суммы ряда и проведен анализ его эффективности.

**Код программы.**

#include "mpi.h"

#include <iostream>

#include <math.h>

#include <ctime>

#define \_USE\_MATH\_DEFINES

using namespace std;

int main(int argc, char \*argv[]) {

long long N;

double t\_start, t\_end;

double s = 0.0, s\_temp = 0.0;

int i, rank, size;

MPI\_Status stat;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

t\_start = MPI\_Wtime();

if (rank == 0) {

N = stoi(argv[argc-1]);

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

MPI\_Send(&N, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

}

else {

MPI\_Recv(&N, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, &stat);

}

long long h = N / size;

long long m = N % size;

long long begin;

if (rank < m) {

h += 1;

begin = rank \* h;

}

else begin = rank \* h + m;

for (long long i = begin; i < h + begin; i += 2) {

if(i != 1 and i != 0)

s += 1. / (double)(i \* (i - 1));

}

for (long long i = begin + 1; i < h + begin; i += 2) {

if(i != 1 and i != 0)

s -= 1. / (double)(i \* (i - 1));

}

if (rank == 0) {

for (i = 1; i < size; i++) {

MPI\_Recv(&s\_temp, 1, MPI\_DOUBLE, i, 2, MPI\_COMM\_WORLD, &stat);

s += s\_temp;

}

t\_end = MPI\_Wtime();

cout << "TIME = " << t\_end - t\_start << endl;

cout << "SUM = " << s << endl;

}

else {

MPI\_Send(&s, 1, MPI\_DOUBLE, 0, 2, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}