МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №7 по дисциплине «Параллельное программирование»

**Коллективные обмены в MPI**

|  |  |
| --- | --- |
| Выполнил: студент группы ФИб-4301-51-00 | / Д.А. Лаптев / |
| Проверил: ст. преподаватель каф. ПМиИ | / В.А. Бызов / |

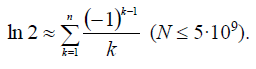
Киров 2021

# **Цель работы**

**Цель работы:** Получить навыки разработки параллельных программ с использованием коллективных обменов MPI.

**Задачи:**

1. Написать параллельную программу, вычисляющую ln 2 по формуле



Реализовать а) с использованием парных операций,

б) с использованием коллективных операций.

Замерить среднее время выполнения программы для n = 108, 5∙108, 109 (на каждое – не менее 3 запусков) для 1, 2, 4 и 8 процессов. Вычислить среднее ускорение для 2, 4 и 8 процессов. Рассчитать эффективность параллельного алгоритма. Построить графики ускорения и эффективности. Сравнить результаты использования парных и коллективных операций.

1. Написать параллельную программу, проверяющую, является ли заданный массив размера N (N <= 5∙108) упорядоченным. Замерить среднее время выполнения программы для N = 108, 5∙108 (на каждое – не менее 3 запусков) для 1, 2, 4 и 8 процессов. Вычислить среднее ускорение для 2, 4 и 8 процессов. Рассчитать эффективность параллельного алгоритма. Построить графики ускорения и эффективности.
2. Написать параллельную программу, вычисляющую произведение квадратной матрицы размера N x N на вектор размера N (N <= 10000). Замерить среднее время выполнения программы для N = 3000, 5000 и 10000 (на каждое – не менее 3 запусков) для 1, 4 и 16 процессов. Вычислить среднее ускорение для 4 и 16 процессов. Рассчитать эффективность параллельного алгоритма. Построить графики ускорения и эффективности.
3. Написать параллельную программу, в которой процессы с чётными номерами вычисляют минимальное значение в массиве из M элементов (M <= 5∙108), а процессы с нечётными номерами вычисляют скалярное произведение векторов размера N (N <= 5∙108). Числа M и N вводятся пользователем. Продемонстрировать корректность работы программы

Задание №1.

Листинг:

#include <iostream>

#include <cmath>

#include <mpi.h>

using namespace std;

int proc\_rank, proc\_num;

void func1(int argc, char\* argv[])

{

const int buf\_size = 1;

int n = 100000000;

//int n = 500000000;

//int n = 1000000000;

MPI\_Status st;

double start, end, diff;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);

if (proc\_rank == 0) {

start = MPI\_Wtime();

}

double sum = 0;

for (int i = proc\_rank; i < n; i += proc\_num)

{

sum += pow(-1, i) / (i + 1);

}

if (proc\_rank == 0) {

for (int i = 1; i < proc\_num; i++)

{

double sum1;

MPI\_Recv(&sum1, 1, MPI\_DOUBLE, i, i, MPI\_COMM\_WORLD, &st);

sum += sum1;

}

}

else

{

MPI\_Send(&sum, 1, MPI\_DOUBLE, 0, proc\_rank, MPI\_COMM\_WORLD);

}

if (proc\_rank == 0)

{

cout << "Task " << proc\_rank << " has end" << " sum=" << sum << endl;

end = MPI\_Wtime();

diff = end - start;

cout << diff << endl;

}

MPI\_Finalize();

}

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

{

func1(argc, argv);

}

Коллективные

#include "pch.h"

#include <iostream>

#include <cmath>

#include <mpi.h>

using namespace std;

int proc\_rank, proc\_num;

void func1(int argc, char\* argv[])

{

const int buf\_size = 1;

int n = 100000000;

//int n = 500000000;

//int n = 1000000000;

double start, end, diff;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);

if (proc\_rank == 0) {

start = MPI\_Wtime();

}

double sum = 0;

for (int i = proc\_rank; i < n; i += proc\_num)

{

sum += pow(-1, i) / (i + 1);

}

double sum1 = 0;

MPI\_Reduce(&sum, &sum1, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (proc\_rank == 0)

{

cout << "Task " << proc\_rank << " has end" << " sum=" << sum1 << endl;

end = MPI\_Wtime();

diff = end - start;

cout << diff << endl;

}

MPI\_Finalize();

}

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

{

func1(argc, argv);

}



Парные обмены

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 4 | 8 |
| 100000000 | 0,490 | 0,268 | 0,157 | 0,121 |
| 500000000 | 2,401 | 1,267 | 0,862 | 0,516 |
| 1000000000 | 4,745 | 2,548 | 1,577 | 1,052 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 | 4 | 8 |
| 100000000 | 1,828 | 3,121 | 4,049 |
| 500000000 | 1,895 | 2,785 | 4,653 |
| 1000000000 | 1,862 | 3,009 | 4,510 |

Коллективные обмены

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 4 | 8 |
| 100000000 | 0,484 | 0,252 | 0,187 | 0,106 |
| 500000000 | 2,441 | 1,346 | 0,839 | 0,520 |
| 1000000000 | 4,831 | 2,551 | 1,626 | 1,059 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 | 4 | 8 |
| 100000000 | 1,920 | 2,588 | 4,566 |
| 500000000 | 1,813 | 2,909 | 4,694 |
| 1000000000 | 1,894 | 2,971 | 4,561 |

Задание №2

Листинг:

#include <iostream>

#include <cmath>

#include <mpi.h>

using namespace std;

int proc\_rank, proc\_num;

int\* createMass(int n)

{

int\* x = new int[n];

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

{

x[i] = i;

//x[i] = -i;

}

return x;

}

void func1(int argc, char\* argv[])

{

const int buf\_size = 1;

int n = 100000000;

//int n = 500000000;

int\* x=new int[1], \* z;

bool\* res;

double start, end, diff;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);

if (proc\_rank == 0)

{

x = createMass(n);

start = MPI\_Wtime();

}

int\* y=new int[n/proc\_num];

MPI\_Scatter(x, n / proc\_num, MPI\_INTEGER, y, n / proc\_num, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

int last;

bool b = true;

for (int i = 1; i < n / proc\_num; i++)

{

if (y[i] < y[i - 1]) {

b = false;

}

}

last = y[n / proc\_num - 1];

z = new int[proc\_num];

MPI\_Gather(&last, 1, MPI\_INTEGER, z, 1, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

if (proc\_rank == 0)

{

for (int i = 1; i < proc\_num; i++)

{

if (z[i] < z[i - 1])

{

b = false;

}

}

}

res = new bool[proc\_num];

MPI\_Gather(&b, 1, MPI\_C\_BOOL, res, 1, MPI\_C\_BOOL, 0, MPI\_COMM\_WORLD);

if (proc\_rank == 0)

{

bool result = true;

for (int i = 1; i < proc\_num; i++)

{

if (!res[i]) {

result = false;

}

}

cout << "Task " << proc\_rank << " has end" << " sorted=" << result << endl;

end = MPI\_Wtime();

diff = end - start;

cout << diff << endl;

}

MPI\_Finalize();

delete[] x;

delete[] y;

delete[] res;

delete[] z;

}

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

{

func1(argc, argv);

}



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 4 | 8 |
| 100000000 | 0,180 | 0,123 | 0,133 | 0,138 |
| 500000000 | 0,880 | 0,580 | 0,716 | 0,754 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 | 4 | 8 |
| 100000000 | 1,463 | 1,353 | 1,304 |
| 500000000 | 1,517 | 1,229 | 1,167 |

Задание №3

#include <iostream>

#include <cmath>

#include <mpi.h>

using namespace std;

int proc\_rank, proc\_num;

int\* functionCreate1(int n)

{

int\* x = new int[n \* n];

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

{

for (int j = 0; j < n; j++)

{

x[i \* n + j] = 1;

}

}

return x;

}

int\* functionCreate2(int n)

{

int\* x = new int[n];

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

{

x[i] = 1;

}

return x;

}

void func1(int argc, char\* argv[])

{

const int buf\_size = 1;

int n = 3000;

//int n = 5000;

//int n = 10000;

int\* x = new int[1], \* y = new int[n], \* z;

double start, end, diff;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);

if (proc\_rank == 0)

{

delete[] x;

delete[] y;

x = functionCreate1(n);

y = functionCreate2(n);

start = MPI\_Wtime();

}

int\* countsx = new int[proc\_num];

int\* countsrx1 = new int[proc\_num];

double k = (double)n / proc\_num;

int k1 = (int)k;

if (n%proc\_num!=0)

k1 = n / proc\_num + 1;

for (int i = 0; i < proc\_num; i++)

{

countsx[i] = min((int)((n - k1 \* i) \* n), (int)(k1 \* n));

countsrx1[i] = i \* n \* k1;

}

int\* countsy = new int[proc\_num];

int\* countsry1 = new int[proc\_num];

for (int i = 0; i < proc\_num; i++)

{

countsy[i] = min((int)((n - k1 \* i)), (int)(k1));

countsry1[i] = i \* k1;

}

int\* x1 = new int[countsx[proc\_rank]];

MPI\_Scatterv(x, countsx, countsrx1, MPI\_INTEGER, x1, countsx[proc\_rank], MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

MPI\_Bcast(y, n, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

int\* sum = new int[countsy[proc\_rank]];

for (int i = 0; i < countsy[proc\_rank]; i++)

{

sum[i] = 0;

for (int j = 0; j < n; j++) {

sum[i] += x1[i \* n + j] \* y[j];

}

}

z = new int[n];

MPI\_Gatherv(sum, countsy[proc\_rank], MPI\_INTEGER, z, countsy, countsry1, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

if (proc\_rank == 0)

{

end = MPI\_Wtime();

diff = end - start;

cout << diff << endl;

}

MPI\_Finalize();

delete[] x;

delete[] x1;

delete[] y;

delete[] z;

delete[] sum;

}

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

{

func1(argc, argv);

}

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 4 | 16 |
| 3000 | 0,007 | 0,012 | 0,016 |
| 5000 | 0,027 | 0,027 | 0,035 |
| 10000 | 0,078 | 0,118 | 0,124 |

|  |  |  |
| --- | --- | --- |
|  | 4 | 16 |
| 3000 | 0,583 | 0,438 |
| 5000 | 1,000 | 0,771 |
| 10000 | 0,661 | 0,629 |

Задание №4

Листинг:

#include <iostream>

#include <cmath>

#include <mpi.h>

using namespace std;

int proc\_rank, proc\_num;

int\* functionCreate1(int n)

{

int\* x = new int[n];

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

{

x[i] = i;

}

return x;

}

void func1(int argc, char\* argv[])

{

const int buf\_size = 1;

int n = 0, m = 0;

int\* x = new int[1], \* y = new int[1], \* z = new int[1];

double start, end, diff;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);

if (proc\_rank == 0)

{

cout << "Input n" << endl;

cin >> n;

cout << "Input m" << endl;

cin >> m;

if (n < 0)

n = 0;

if (n > 500000000)

n = 500000000;

if (m < 0)

m = 0;

if (m > 500000000)

m = 500000000;

x = functionCreate1(n);

y = functionCreate1(n);

z = functionCreate1(m);

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

cout << x[i] << ' ';

}

cout << endl;

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

cout << y[i] << ' ';

}

cout << endl;

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

cout << z[i] << ' ';

}

cout << endl;

start = MPI\_Wtime();

}

MPI\_Bcast(&n, 1, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

MPI\_Bcast(&m, 1, MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

int\* countsn = new int[proc\_num];

int\* countsrn1 = new int[proc\_num];

double kn = (double)n \* 2 / proc\_num;

int kn1 = (int)kn;

double km = (double)m \* 2 / proc\_num;

int km1 = (int)km;

if (n \* 2 % proc\_num != 0)

kn1 = n \* 2 / proc\_num + 1;

if (m \* 2 % proc\_num != 0)

km1 = m \* 2 / proc\_num + 1;

for (int i = 0; i < proc\_num; i++)

{

if (i % 2 == 0)

{

countsn[i] = 0;

countsrn1[i] = 0;

}

else

{

countsn[i] = min((int)((n - kn1 \* (i / 2))), (int)(kn1));

countsrn1[i] = i / 2 \* kn1;

}

}

int\* countsm = new int[proc\_num];

int\* countsrm1 = new int[proc\_num];

for (int i = 0; i < proc\_num; i++)

{

if (i % 2 != 0)

{

countsm[i] = 0;

countsrm1[i] = 0;

}

else

{

countsm[i] = min((int)((n - km1 \* (i / 2))), (int)(km1));

countsrm1[i] = i / 2 \* km1;

}

}

int\* x1 = new int[countsn[proc\_rank]];

int\* y1 = new int[countsn[proc\_rank]];

int\* z1 = new int[countsm[proc\_rank]];

MPI\_Scatterv(x, countsn, countsrn1, MPI\_INTEGER, x1, countsn[proc\_rank], MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

MPI\_Scatterv(y, countsn, countsrn1, MPI\_INTEGER, y1, countsn[proc\_rank], MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

MPI\_Scatterv(z, countsm, countsrm1, MPI\_INTEGER, z1, countsm[proc\_rank], MPI\_INTEGER, 0, MPI\_COMM\_WORLD);

int Min = 1000000000, sum = 0;

if (proc\_rank % 2 == 0)

{

for (int i = 0; i < countsm[proc\_rank]; i++) {

Min = min(z1[i], Min);

}

}

else

{

for (int i = 0; i < countsn[proc\_rank]; i++) {

sum += x1[i] \* y1[i];

}

}

int Min1, sum1;

MPI\_Reduce(&Min, &Min1, 1, MPI\_INTEGER, MPI\_MIN, 0, MPI\_COMM\_WORLD);

MPI\_Reduce(&sum, &sum1, 1, MPI\_INTEGER, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (proc\_rank == 0)

{

cout << "SUM = " << sum1 << endl;

cout << "MIN = " << Min1 << endl;

end = MPI\_Wtime();

diff = end - start;

cout << diff << endl;

}

MPI\_Finalize();

delete[] x;

delete[] x1;

delete[] y1;

delete[] z1;

delete[] y;

delete[] z;

}

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

func1(argc, argv);

}

Изображение выглядит как текст

Автоматически созданное описание

Вывод

В результате выполнения лабораторной работы были получены навыки работы с коллективных взаимодействием технологии MPI и были реализованы задачи с применением MPI.