Национальный технический университет Украины

«Киевский политехнический институт»

Лабораторная работа №8

По курсу

Параллельные и распределенные вычисления

Тема:

MPI

Работу выполнил

Студент 3-го курса ФИВТ

Группы ИВ-73

Грубый Павел

Киев-2010

Техническое задание



A = B×(MX×MZ - a×ME)

Этап 1. Построение параллельного алгоритма

AH = B × ( MX×MZH - a×MEH)

Этап 2. Разработка алгоритмов работы каждого процесса

**Алгоритм верхней линии**

1. Если задача крайняя правая – ввести B, ME, иначе принять B, ME(X+1)H от задачи справа.
2. Передать B, MEXH задаче слева
3. Передать B, MEУH задаче снизу
4. Принять MZH, A, MX от задача снизу
5. Счет AH = B × ( MX×MZH - a×MEH)
6. Передать AH задаче снизу

**Алгоритм средней линии**

1. Принять B, MEXH от задачи сверху
2. Передать B, MEУН задаче снизу
3. Принять MZH и A, MX от задачи снизу
4. Передать MZH и A, MX задаче сверху
5. Счет AH = B × ( MX×MZH - a×MEH)
6. Принять AXH от задачи сверху
7. Передать AУH задаче снизу

**Алгоритм нижней линии**

1. Принять B, MEH от задачи сверху.

Если задача крайняя левая:

1. Ввести MВ
2. Передать MZH задаче справа и задаче сверху
3. Принять A, MX от задачи слева

Иначе, если задача крайняя правая:

1. Принять MZH от задачи слева
2. Передать MZH задаче сверху
3. Ввести A, MX
4. Передать A, MX задаче слева и задаче сверху

Иначе:

1. Принять MZH от задачи слева
2. Переслать MZH задаче справа и задаче сверху
3. Принять A, MX от задачи справа
4. Передать A, MX задаче слева

Всем:

1. Передать A, MX задаче сверху
2. Счет AH = B × ( MX×MZH - a×MEH)
3. Принять AХН от задачи сверху

Если задача крайняя правая:

1. Передать AХН задаче слева

Иначе:

1. Принять AУН от задачи справа

Если задача не крайняя левая:

1. Передать AZН от задаче слева

Иначе:

Вывод МА

Этап 3. Разработка структурной схемы взаимодействия задач



рис. 3.1. Схема взаимодействия задач.

Этап 4. Разработка программы

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Parallel and distributed calculations

// Laboratory Work # 8

// MPI

//

// MA = B \* (MX \* MZ - a \* ME)

//

// Grooby Pavel

// IO-73

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdafx.h"

#include <mpi.h>

#include "windows.h"

#include <iostream>

#include <fstream>

#include <math.h>

using namespace std;

int N;

int k;

int p;

int H;

int el = 1;

int getMElement(int\* matr, int i, int j){

return matr[i\*N+j];

}

int getVElement(int\* vec, int i){

return vec[i];

}

void setMElement(int\* matr, int i, int j, int val){

matr[i\*N+j]=val;

}

void setVElement(int\* vec, int i, int val){

vec[i]=val;

}

void outputMatr(int\* matr,int rank, char n[]){

char buffer[15];

int l = sprintf (buffer, "%d\_%s", rank,n);

ofstream outf(buffer);

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

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

outf << getMElement(matr,i,j) << " ";

}

outf<<"\n";

}

outf.close();

}

void outputVec(int\* vec,int rank, char n[]){

char buffer[15];

//int l = sprintf (buffer, "%d\_%s", rank,n);

printf ("\nT%d RESULT: ", rank);

ofstream outf(buffer);

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

cout << getVElement(vec,i) << " ";

}

outf.close();

}

void inputMatr(int\* matr){

int a = 1;

for (int i = 0; i < N\*N; i++){

matr[i] = a;

//a++;

}

}

void inputVec(int\* vec){

int a = 1;

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

vec[i] = a;

//a++;

}

}

void transMatr(int\* matr){

int buf = 0;

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

for (int j = i; j < N; j++) {

buf = matr[i\*N+j];

matr[i\*N+j] = matr[j\*N+i];

matr[j\*N+i] = buf;

}

}

}

int\* calc (int\* MZ, int\* MX, int\* ME, int\* B, int\* a, int rank){

//MBo\*(MZo\*MX)

int\* M4 = new int[H\*N];

int\* rez = new int[H\*N];

int el = 0;

for (int i = 0; i < H\*N; i++){

M4[i] = 0;

rez[i] = 0;

}

for (int h = 0; h < H; h++){

for (int k = 0; k < N; k++){

el = 0;

for (int es = 0; es < N; es++){

el = el + getMElement(MX,k,es) \* MZ[h\*N+es];

}

M4[h\*N+k] = el - (\*a)\*ME[h\*N+k];

}

for (int k = 0; k < N; k++){

el = 0;

for (int es = 0; es < N; es++){

el = el + getVElement(B,es) \* M4[h\*N+es];

}

rez[h\*N+k] = el;

}

}

return rez;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

{

int rank, size;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

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

if (rank == 0) {

printf("\nN:");fflush(stdout);

scanf\_s("%d",&N);

}

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

p = size;

k = (int)sqrt((double)p);

H = N/p;

//printf ("\nT%d: N = %d and k = %d and H = %d and p = %d", rank, N, k, H, p);

printf ("\nT%d: START ", rank);

int sendMatrLineSize = rank\*k\*H\*N;

int sendMatrColSize = (k - 1)\*H\*N;

int recMatrLineSize = (rank+1)\*k\*H\*N;

int sendVecLineSize = rank\*k\*H;

int sendVecColSize = (k - 1)\*H;

int recVecLineSize = (rank+1)\*k\*H;

int side = rank % k;

//top

if (rank < k) {

int\* ME = new int [N\*N];

int\* MZ = new int [N\*N];

int\* MX = new int [N\*N];

int\* B = new int [N];

int\* a = new int;

if ((rank % k) == (k - 1) ) {//top right corner

inputMatr(ME);

inputVec(B);

//setMElement(ME,2,2,2);

//setVElement(B,1,2);

//transMatr(ME);

//outputMatr(ME,N,rank,"inputMX.txt");

}else{

MPI\_Recv(ME,recMatrLineSize,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(B,N,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

}

if (rank != 0){

//sending to left

MPI\_Send(ME+H\*N\*k,sendMatrLineSize,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

MPI\_Send(B,N,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

}

//sending down

MPI\_Send(ME+H\*N,sendMatrColSize, MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

MPI\_Send(B,N, MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

MPI\_Recv(MZ,N\*N,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MX,H\*N,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(a,1,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

printf ("\nT%d: CALCULATES ", rank);

int\* rez = calc (MZ, MX, ME, B, a, rank);

MPI\_Send(rez,H, MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

delete ME;

delete MZ;

delete MX;

delete B;

delete a;

delete rez;

}

else

//down

if (rank >= p - k){

int\* ME = new int [N\*N];

int\* MZ = new int [N\*N];

int\* MX = new int [N\*N];

int\* B = new int [N];

int\* a = new int;

MPI\_Recv(ME,N\*H,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(B,N,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

if (rank == (k\*k - k) ) {//down left corner

inputMatr(MZ);

MPI\_Send(MZ+H\*N\*k,sendMatrLineSize,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD);

MPI\_Send(MZ+H\*N,sendMatrColSize,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

MPI\_Recv(MX,N\*N,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(a,1,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

}else {

if ( rank == (k\*k - 1) ) {//down right corner

MPI\_Recv(MZ,recMatrLineSize,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD,&status);

MPI\_Send(MZ+H\*N,sendMatrColSize,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

inputMatr(MX);

\*a = 1;

MPI\_Send(MX,N\*N,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

MPI\_Send(a,1,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

}else {//down center

MPI\_Recv(MZ,recMatrLineSize,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD,&status);

MPI\_Send(MZ+H\*N\*k,sendMatrLineSize,rank+1,0,MPI\_COMM\_WORLD);

MPI\_Send(MZ+H\*N,sendMatrColSize,N\*N,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

MPI\_Recv(MX,N\*N,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(a,1,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

MPI\_Send(MX,N\*N,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

MPI\_Send(a,1,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

}

}

MPI\_Send(MX,N\*N,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

MPI\_Send(a,1,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

printf ("\nT%d: CALCULATES ", rank);

int\* rez = calc (MZ, MX, ME, B, a, rank);

int\* drez = new int[k\*H];

MPI\_Recv(drez,(k - 1)\*H,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

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

drez[i+(k-1)\*H] = rez[i];

}

delete rez;

if (rank == (k\*k - 1) ){ //right corner

MPI\_Send(drez,k\*H,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

}else {

if (rank != (k\*k - k) ){ //all except left down

int\* ddrez = new int[(k\*k-rank)\*k\*H];

MPI\_Recv(ddrez,((k\*k-rank)\*k-k)\*H,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

for (int i = 0; i < k\*H; i ++){

ddrez[i+((k\*k-rank)\*k-k)\*H] = drez[i];

}

MPI\_Send(ddrez,(k\*k-rank)\*k\*H,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD);

delete ddrez;

}else{//left down

int\* lrez = new int[N];

MPI\_Recv(lrez, (k\*k - k)\*H,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD,&status);

for (int i = 0; i < k\*H; i ++){

lrez[i+(k\*k - k)\*H] = drez[i];

}

outputVec(lrez,rank,"outA.txt");

delete ME;

delete MZ;

delete MX;

delete lrez;

delete drez;

}

}

}

//center

else{

div\_t divr = div(rank, k);

int l = divr.quot;

int centRecSize = (k - l)\*H\*N;

int centSendSize = (k - l - 1)\*H\*N;

int\* ME = new int [N\*N];

int\* MZ = new int [N\*N];

int\* MX = new int [N\*N];

int\* B = new int [N];

int\* a = new int;

MPI\_Recv(ME,centRecSize,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(B,N,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

MPI\_Send(ME+N\*H,centSendSize,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

MPI\_Send(B,N,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

MPI\_Recv(MZ,centRecSize,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(MX,N\*N,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

MPI\_Recv(a,1,MPI\_INT,rank+k,0,MPI\_COMM\_WORLD,&status);

MPI\_Send(MZ+N\*H,centSendSize,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

MPI\_Send(MX,N\*N,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

MPI\_Send(a,1,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD);

printf ("\nT%d: CALCULATES ", rank);

int\* rez = calc (MZ, MX, ME, B, a, rank);

int centDownRSize = l\*H;

int centDownSSize = (l+1)\*H;

int\* drez = new int[centDownSSize];

MPI\_Recv(drez,centDownRSize,MPI\_INT,rank-k,0,MPI\_COMM\_WORLD,&status);

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

drez[i+H\*l] = rez[i];

}

MPI\_Send(drez,centDownSSize, MPI\_INT,rank+k,0,MPI\_COMM\_WORLD);

delete ME;

delete MZ;

delete MX;

delete rez;

delete drez;

}

printf ("\nT%d: END\n", rank);

MPI\_Finalize();

return 0;

}