Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

 «Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

**ОТЧЕТ**

по лабораторной работе №4

на тему: «Параллельное блочное умножение матриц с помощью MPI»

Выполнил: студент группы РИС-20-1б

Шумилов Л.С.

Проверил: доцент кафедры ИТАС

Щапов В.А.

Пермь, 2024

**Цель работы** – реализовать алгоритм параллельного умножения матрицы блочным методом

**Задание**:

1. Установить OpenMPI;
2. Реализовать алгоритм параллельного умножения матриц блочным методом.

**Ход работы**

Блочный метод умножения матриц — это метод, который позволяет умножать матрицы на блоки, разделяя их на более мелкие подматрицы. Этот метод основан на стандартном способе умножения матриц, но вместо того, чтобы перемножать отдельные элементы, мы перемножаем блоки матриц.

Преимуществом блочного метода умножения матриц является то, что он может быть выполнен на параллельных вычислительных системах, таких как многопроцессорные и многоядерные процессоры. Это позволяет ускорить вычисления и сделать их более эффективными.

Кроме того, блочный метод умножения матриц может быть использован для умножения очень больших матриц, которые не могут быть размещены в памяти одного компьютера. В этом случае, матрицы могут быть разделены на блоки и перемножены на разных узлах распределённой вычислительной системы.

Для реализации параллельных операций применяется решение OpenMPI для языка программирования C++, который позволит распараллелить вычисления.

На Листинге 1 представлен метод умножения внутри 1 блока матрицы

Листинг 1. Перемножение внутри блока

void block\_multiplication(int\* matrix\_a, int\* matrix\_b, int\* result\_matrix, int block\_size) {

for (int i = 0; i < block\_size; i += 1) {

for (int k = 0; k < block\_size; k += 1) {

for (int j = 0; j < block\_size; j += 1) {

int row\_position = i \* SIZE;

result\_matrix[row\_position + j] = matrix\_a[row\_position + k] + matrix\_b[row\_position + j];

}

}

}

}

На Листинге 2 представлен пример инициализации OpenMPI для начала работы с данным инструментом в программе.

Листинг 2. Инициализация OpenMPI в программе

void mpi\_thread\_init(int\* rank, int\* thread\_count) {

MPI\_Init(NULL, NULL);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, thread\_count);

}

Следует отметить, что OpenMPI работает с цельными участками памяти, поэтому матрицы объявляются внутри одномерных динамических массивов, чтобы система могла обработать 1 непрерывный участок памяти.

Листинг 3. Инициализация матриц

int rank, thread\_count;

mpi\_thread\_init(&rank, &thread\_count);

double start\_time;

int \*matrix\_a = new int[SIZE \* SIZE],

\*matrix\_b = new int[SIZE \* SIZE],

\*out\_matrix = new int[SIZE \* SIZE];

init\_in\_first\_rank(rank, matrix\_a, matrix\_b, out\_matrix, start\_time);

int full\_size, block\_size, row\_block\_size, block\_count;

compute\_sizes(thread\_count, &full\_size, &block\_size, &row\_block\_size, &block\_count);

В Листинге 4 представлен сам процесс параллельного умножения матриц и вызов блочной функции перемножения.

Листинг 4. Параллельное умножение матриц

int int root = 0;

MPI\_Bcast(matrix\_b, full\_size, MPI\_INT, root, MPI\_COMM\_WORLD);

int\* mpi\_block\_matrix = new int[block\_size];

int\* mpi\_result\_block\_matrix = new int[block\_size];

for (int i = 0; i < block\_size; i += 1) {

mpi\_result\_block\_matrix[i] = 0;

}

MPI\_Scatter(matrix\_a, block\_size, MPI\_INT, mpi\_block\_matrix, block\_size, MPI\_INT, root, MPI\_COMM\_WORLD);

block\_multiplication(mpi\_block\_matrix, matrix\_b, mpi\_result\_block\_matrix, row\_block\_size);

MPI\_Gather(mpi\_result\_block\_matrix, block\_size, MPI\_INT, out\_matrix, block\_size, MPI\_INT, root, MPI\_COMM\_WORLD);

show\_endtime\_first\_rank(rank, start\_time);

На Рисунке 1 показан результат выполнения программы с замером времени на умножение матриц размером 2000 на 2000 каждая.



Рисунок 1. Время умножения матриц