**Практическая работа №5**

**Вариант №3.7**

**Формулировка задания.**

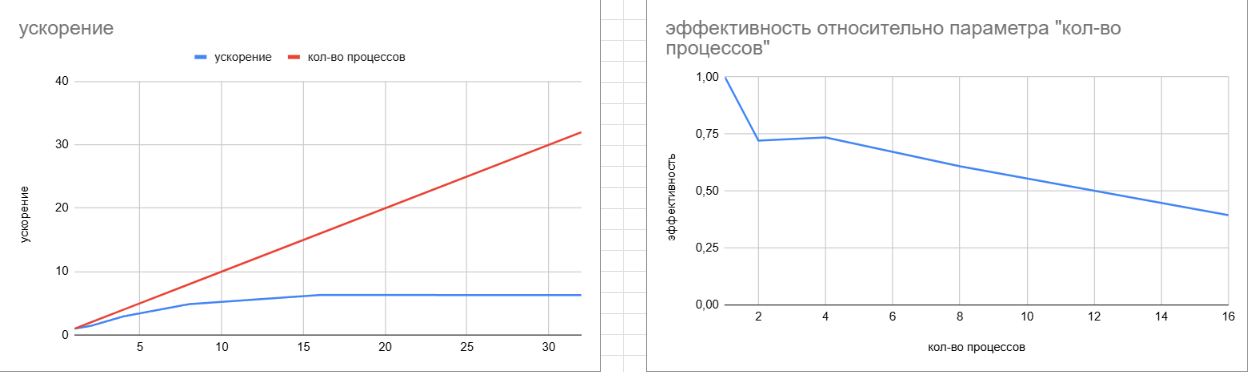
Исследовать время счета, ускорение и эффективность OpenMP программы умножения матриц для вариантов циклов i-j-k и k-i-j. С помощью ключей static для директивы for найти распределение итераций по нитям с минимальным временем выполнения. Оценку времени работы программы проводить минимум по трем ее запускам. Обосновать полученные результаты.

**Ход работы.**

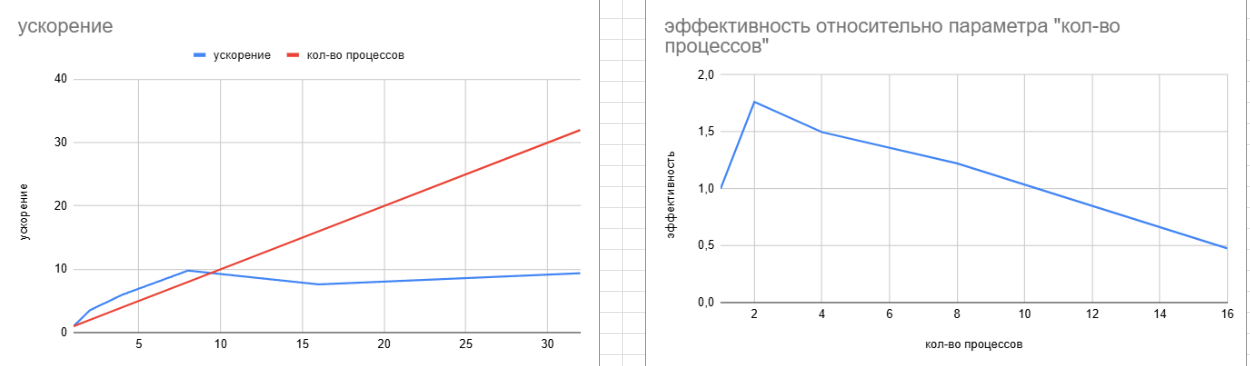
В самом начале мы создаем массивы состоящий из случайных чисел в диапазоне от [0, 1] . После чего применяя директивы #pragma omp parallel for даем доступ другим потокам к созданным массивам, используем ключ collapse(3) для объединения 3 вложенных циклов в один параллельный цикл, применяем ключ schedule(static) для статического распределения итераций по нитям.

График ускорение и эффективности i-j-k

|  |  |  |  |
| --- | --- | --- | --- |
| кол-во процессов | время в мкс | ускорение | эффективность |
| 1 | 4403 | 1 | 1 |
| 2 | 3053 | 1,442188012 | 0,7210940059 |
| 4 | 1498 | 2,939252336 | 0,7348130841 |
| 8 | 905 | 4,86519337 | 0,6081491713 |
| 16 | 698 | 6,308022923 | 0,3942514327 |
| 32 | 699 | 6,298998569 | 0,1968437053 |



|  |  |  |  |
| --- | --- | --- | --- |
| кол-во процессов | время в мкс | ускорение | эффективность |
| 1 | 2085 | 1 | 1 |
| 2 | 1249 | 3,525220176 | 1,762610088 |
| 4 | 735 | 5,99047619 | 1,497619048 |
| 8 | 450 | 9,784444444 | 1,223055556 |
| 16 | 579 | 7,604490501 | 0,4752806563 |
| 32 | 470 | 9,368085106 | 0,2927526596 |

****

**Заключение**

В данной работе была написана параллельная программа для умножения матриц путем использования циклов i-j-k и k-i-j. По итогам, мы видим, что k-i-j проходит быстрее нежели, i-j-k. Данный процесс происходит по причине того, что k-i-j порядок более эффективен для использования кэша процессора, когда используется порядок циклов k-i-j, данные из матрицы B, которые используются внутри цикла по i, остаются в кэше процессора на протяжении всего цикла по j.

**Код программы**

#include <iostream>

#include <stdlib.h>

#include <cmath>

#include <omp.h>

#include <random>

#include <ctime>

#define N 1000

#define SIZE 32

void fillMatrix(double\*\* A, int size) {

using namespace std;

random\_device rd;

mt19937 gen(rd());

uniform\_real\_distribution<double> dis(0.0, 1.0);

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

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

A[i][j] = dis(gen);

}

}

}

int main()

{

omp\_set\_num\_threads(SIZE);

double\*\* A = new double\*[N];

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

A[i] = new double[N];

}

double\*\* B = new double\* [N];

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

B[i] = new double[N];

}

double\*\* C = new double\* [N];

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

C[i] = new double[N];

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

C[i][j] = 0.0;

}

}

// Инициализация матриц A и B

fillMatrix(A, N);

fillMatrix(B, N);

// Вариант циклов i-j-k

clock\_t start\_time = clock();

#pragma omp parallel for shared(A, B, C) collapse(3) schedule(static)

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

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

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

C[i][j] += A[i][k] \* B[k][j];

}

}

}

std::cout << "Time for i-j-k loop: " << (double)(clock() - start\_time) << std::endl;

// Обнуление матрицы C для следующего варианта циклов

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

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

C[i][j] = 0.0;

}

}

// Вариант циклов k-i-j

start\_time = clock();

#pragma omp parallel for shared(A, B, C) collapse(3) schedule(static)

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

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

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

C[i][j] += A[i][k] \* B[k][j];

}

}

}

std::cout << "Time for k-i-j loop: " << (double)(clock() - start\_time) << std::endl;

return 0;

}