МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4
«OpenMP»

Лабораторная работа
по дисциплине
«Параллельные и распределенные вычисления»
студента 4 курса группы ПИ-б-о-182(2)
Змитрович Никита Сергеевич

направления подготовки 09.03.04 «Программная инженерия»

Научный руководитель
старший преподаватель кафедры
компьютерной инженерии и моделирования

_____
(оценка)

_____
(подпись, дата)

Чабанов В.В.

Симферополь, 2021

Цель:

1. Изучить средства OpenMP предназначенные для создания многопоточных программ для систем с общей памятью;
2. Реализовать приложение выполняющее многопоточные вычисления;
3. Сравнить скорость выполнения вычислений при условии использования различного количества потоков.

Постановка задачи:

Даны две квадратные матрицы A и B вещественных чисел. Получите матрицу C, которая является произведением матриц A и B. Постройте зависимость ряд зависимостей времени решения задачи от количества использованных потоков.

Выполнение работы

График 1 — Зависимость времени решения от количества потоков для n = 500

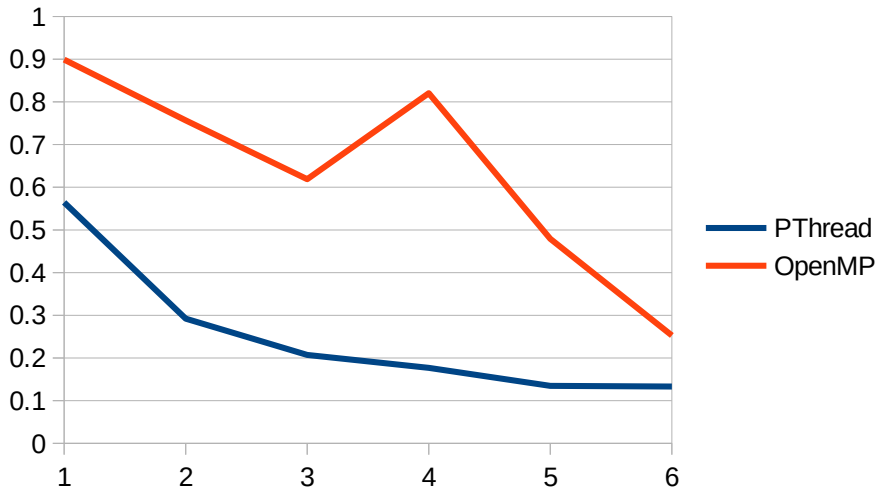


График 2 — Зависимость времени решения от количества потоков для n = 1000

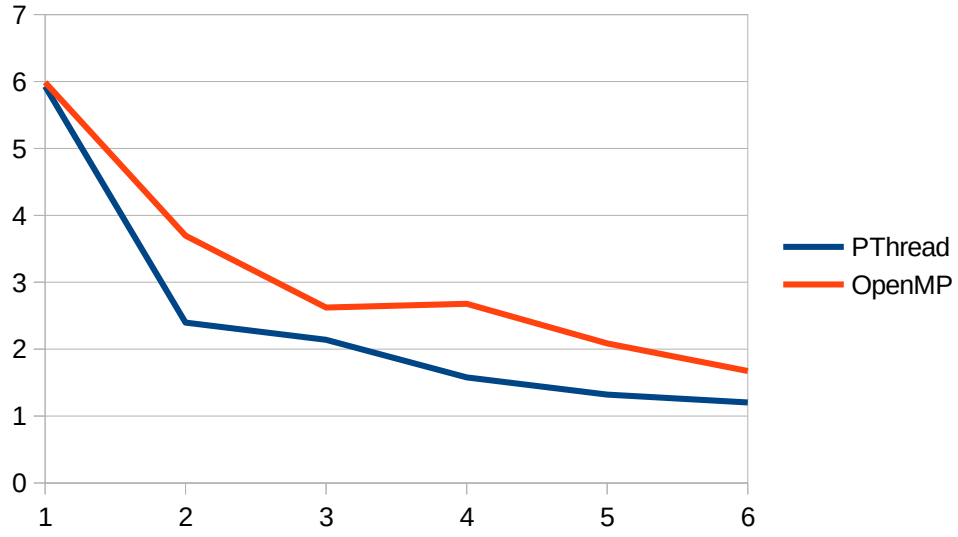
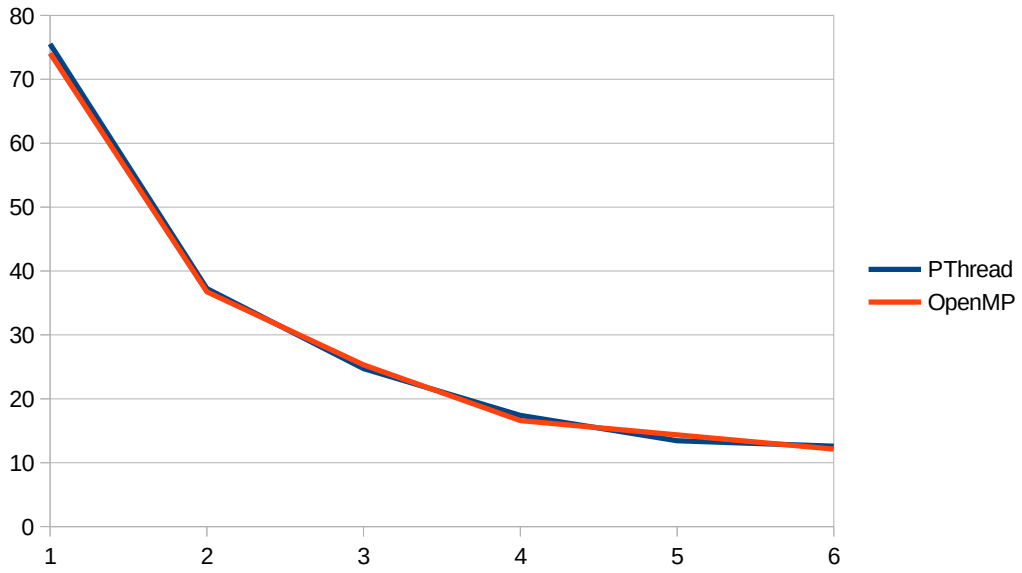


График 3 — Зависимость времени решения от количества потоков для n = 2000



Вывод:

OpenMP медленнее PThread с маленьким объёмом данных, это видно из графика 1 и 2. Однако для n = 2000 OpenMP сравнялся по скорости PThread. Также у OpenMP есть одно большое преимущество, он намного проще позволяет распределять вычисления между разными потоками.

Приложение

```
#include <iostream>
#include <omp.h>

int main()
{
int n = 0, available_threads = 0;

std::cout << "Enter number row/columns" << std::endl;
std::cin >> n;

std::cout << omp_get_num_procs() << " available threads" << std::endl;
std::cout << "Choose number of threads: ";
std::cin >> available_threads;
std::cout << std::endl;

omp_set_num_threads(available_threads);

int** A = new int*[n];
int** B = new int*[n];
int** C = new int*[n];

for (int i = 0; i < n; i++)
{
A[i] = new int[n];
B[i] = new int[n];
C[i] = new int[n];

for (int j = 0; j < n; j++)
{
A[i][j] = random() % 10;
B[i][j] = random() % 10;
C[i][j] = 0;
}
}

int start = omp_get_wtime();

#pragma omp parallel for
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 << "Ellapsed " << omp_get_wtime() - start << " seconds";
return 0;
}
```