МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 1

по курсу   
Параллельное программирование

Группа 6409

Студент П.А. Маштаков

(*подпись*)

Преподаватель

В.Д. Зайцев

(*подпись*)

Самара 2023

ЗАДАНИЕ

**Произвести запуск программ** «Hello world»**, которые используют технологии MPI и OpenMP, на различном количестве процессоров (потоков). В ходе анализа работы программы оценить время ее выполнения на различном количестве исполняющих нитей (процессов), для чего в заготовках необходимо произвести изменения. Здесь и далее для уменьшения влияния сторонних факторов необходимо проводить усреднение времени выполнения не менее чем по 12 запускам.**

Таблица 1 – Исходные данные на ЛР № 1

|  |  |
| --- | --- |
| Количество процессов/потоков | 4, 8 |

ВВЕДЕНИЕ

Использование параллельных вычислительных систем становится все более актуальным направлением в современном мире. Это обусловлено наличием задач, требующих больших вычислительных ресурсов, для которых существующей вычислительной техники недостаточно. Одними из наиболее популярных технологий параллельного программирования являются технологии MPI и OpenMP [1].

В ходе выполнения лабораторной работы было рассчитано время выполнения программы «Hello, World» при параллельном запуске на заданном количестве процессов и использовании технологий MPI и OpenMP, были проанализированы различия во времени между работой исследуемых технологий и последовательного алгоритма.

ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

2.1 Характеристики ПК

В таблице 2 представлены характеристики ПК.

Таблица 2 – Характеристики ПК

|  |  |
| --- | --- |
| Процессор | Intel Celeron N4000 |
| Количество ядер | 2 |
| Количество потоков | 2 |
| Оперативная память | 8 гб |
| Тип системы | Windows |

2.2 Результаты работы программ

В ходе исследования времени работы программ здесь и далее проводилось усреднение не менее чем по 12 запускам.

На рисунках 1-2 представлены скрины запуска и работы программ.

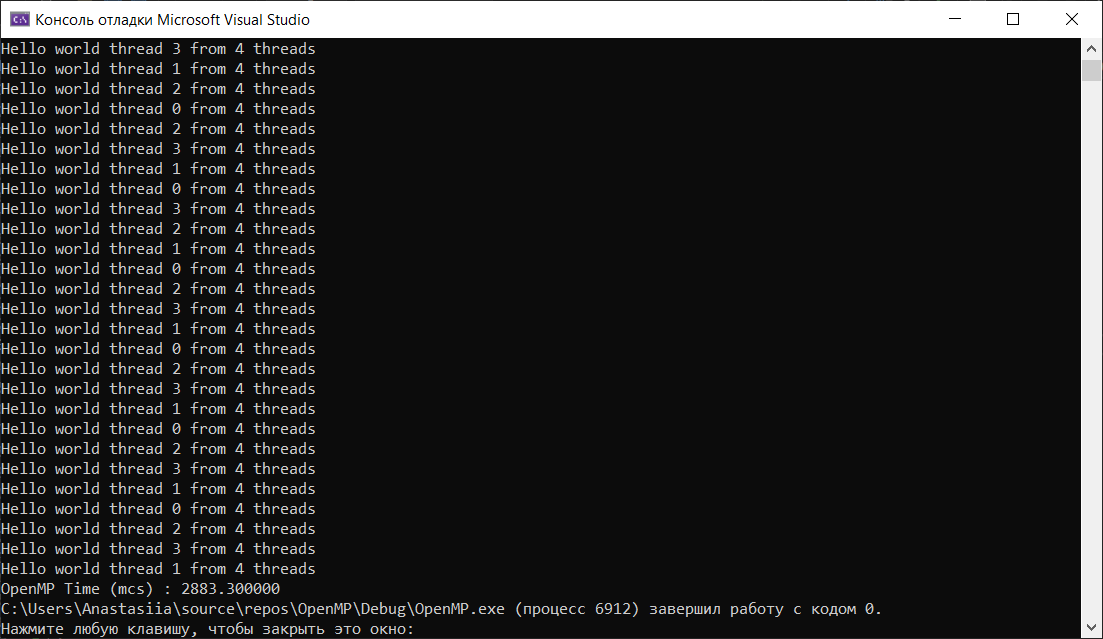


Рисунок 1 – Пример работы программы OpenMP на 4 процессах

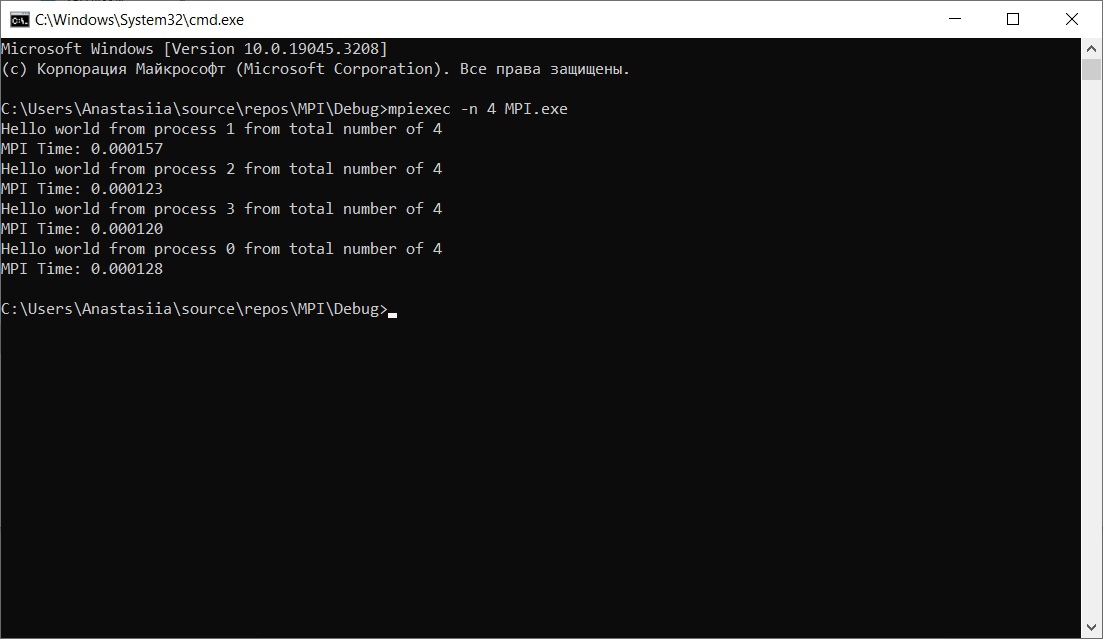


Рисунок 2 – Пример работы программы MPI на 4 процессах

Последовательная программа представляла собой вывод текста на экран. Время работы последовательной программы при четырех выводах текста составило 10081 мкс, при восьми – 20816 мкс.

В таблице 3 представлено время выполнения параллельных программ и их ускорение по сравнению с последовательным вариантом.

Таблица 3 – Результаты выполнения программ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во процессов, шт | OpenMP | | MPI | |
| Время, мкс | Ускорение | Время, мкс | Ускорение |
| 4 | 7622 | 1,3226 | 530 | 19,0207 |
| 8 | 12471 | 1,6691 | 1193 | 17,4484 |

На рисунке 3 приведен график зависимости времени работы программ от количества процессов. На рисунке 4 приведен график зависимости ускорения программ от количества процессов.

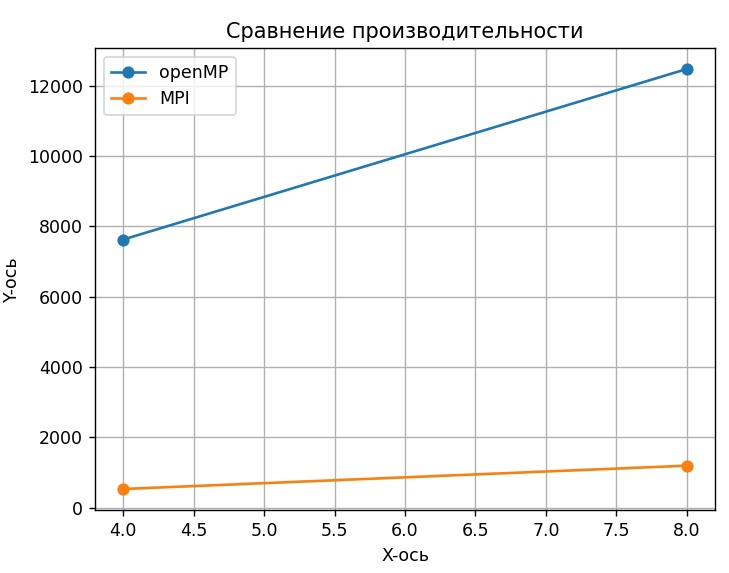


Рисунок 3 – Время работы программ

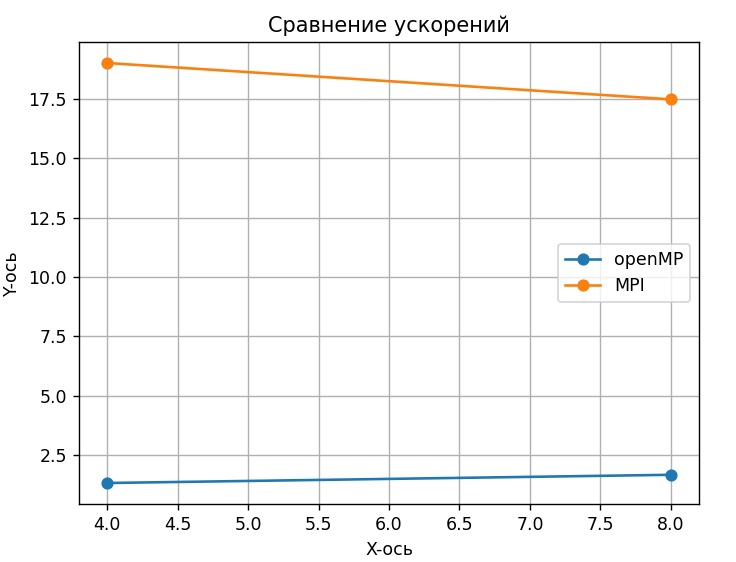


Рисунок 4 – Ускорение программ

**ВЫВОДЫ:**

Из полученных результатов видно, что:

1. При увеличении количества процессов, увеличивается время выполнения параллельных программ. Это может быть связано с тем, что создание процессов занимает время.
2. Максимальное ускорение было получено при использовании технологии MPI на 4 процессах, минимальное – при использовании OpenMP на 4 процессах. Увеличение количества процессов приводит к увеличению времени, которое тратится на их создание.
3. MPI работает быстрее OpenMP. Это можно объяснить тем, что параллельные процессы работают в собственных пространствах памяти.

ЗАКЛЮЧЕНИЕ

Цель лабораторной работы – написать параллельные программы вывода текстового сообщения на экран с использованием технологий MPI и OpenMP и сравнить время выполнения с длительностью последовательной программы достигнута. Показано, что использование параллельных технологий для данного типа программ обосновано, поскольку текст был выведен на экран за меньшее количество времени. Лучше всего себя показала технология MPI. Это можно объяснить тем, что параллельные процессы работают в собственных пространствах памяти.

В ходе выполнения лабораторной работы я изучил основы MPI и OpenMP, приобрел навыки по написанию параллельных программ с использованием вышеперечисленных технологий. Наиболее сложной частью выполнения лабораторной работы была настройка среды разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рыжков, Е. А. Параллельные заметки №1 – технология OpenMP / Е. А. Рыжков // Хабр : [сайт]. – Тула, 2010. – URL: https://habr.com/ru/companies/intel/articles/82486/ (дата обращения: 02.10.2023).
2. Козлова, Е.С. Лабораторные работы по курсу «Параллельное программирование»: Методические указания [Текст] / Сост. Е.С. Козлова, А.С. Широканев − Самара, 2019. – 61 с.
3. Воеводин, В. В. Параллельные вычисления [Текст] / В. В. Воеводин, Вл. В. Воеводин. — СПб.: БХВ-Петербург, 2002. — 608 с.
4. Богачёв К.Ю. Основы параллельного программирования: учебное пособие, 2-е изд. [Текст] / К. Ю. Богачёв ‑ М. : БИНОМ. Лаборатория знаний, 2013. ‑ 344 с.
5. Гергель, В. П. Теория и практика параллельных вычислений, 2-е изд. [Текст] / B. II. Гергель. — М.: Интуит. 2016. - 500 с.
6. Боресков А.В. Параллельные вычисления на GPU. Архитектура и программная модель CUDА Учеб. пособие [Текст] / А.В. Боресков ‑ М.: Издательство Московского университета, 2012. - 336 с.
7. Библиографическое описание документа. Общие требования и правила составления [Электронный ресурс] / сост.: В.С. Крылова, С.М. Григорьевская, Е.Ю. Кичигина // Официальный интернет-сайт научной библиотеки Томского государственного университета. – Электрон. дан. – Томск, [2010]. – <http://www.lib.tsu.ru/win/produkzija/metodichka/metodich.html> (дата обращения: 10.09.2019).

ПРИЛОЖЕНИЕ А

Код программы с технологией MPI

#include "mpi.h"

#include "stdio.h"

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

{

int rank, ranksize, i;

MPI\_Init(&argc, &argv);

double start;

start = MPI\_Wtime();

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

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

printf("Hello world from process %d from total number of %d\n", rank, ranksize);

printf("MPI Time: %f\n", MPI\_Wtime() - start);

MPI\_Finalize();

return 0;

}

ПРИЛОЖЕНИЕ Б

Код программы с технологией OpenMP

#include <omp.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <stdio.h>

#include <cmath>

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

{

double Time = 0;

for (int i = 0; i < 12; ++i) {

omp\_set\_num\_threads(8);

int nThreads, threadNum;

double start = omp\_get\_wtime();

#pragma omp parallel private(nThreads, threadNum)

{

nThreads = omp\_get\_num\_threads();

threadNum = omp\_get\_thread\_num();

printf("Hello world thread %d from %d threads \n", threadNum, nThreads);

}

double end = omp\_get\_wtime();

Time += end - start;

}

printf("OpenMP Time (mcs) : %f ", Time \* (pow(10, 6) / 12));

return 0;

}

ПРИЛОЖЕНИЕ В

Код последовательной программы

#include <iostream>

#include <chrono>

int main() {

auto start\_time = std::chrono::high\_resolution\_clock::now();

for (int i = 0; i < 12; ++i) {

std::cout << "Hello, world!" << std::endl;

}

auto end\_time = std::chrono::high\_resolution\_clock::now();

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(end\_time - start\_time);

std::cout << "Seq. prog. Time: " << duration.count() << " mcs" << std::endl;

return 0;

}