Национальный исследовательский ядерный университет
«МИФИ» (Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

ОТЧЁТ

Лабораторная работа №5:
«Технология MPI. Введение»

Группа Б21-525
Студент Р.Т. Мясников
Преподаватель М.А. Куприяшин

Москва 2023

Оглавление

1. Описание рабочей среды 3
2. Анализ приведенного алгоритма 3
3. Анализ временных характеристик последовательного алгоритма . . . 5
4. Анализ временных характеристик параллельного алгоритма 6
5. Заключение 9
6. Приложение 9

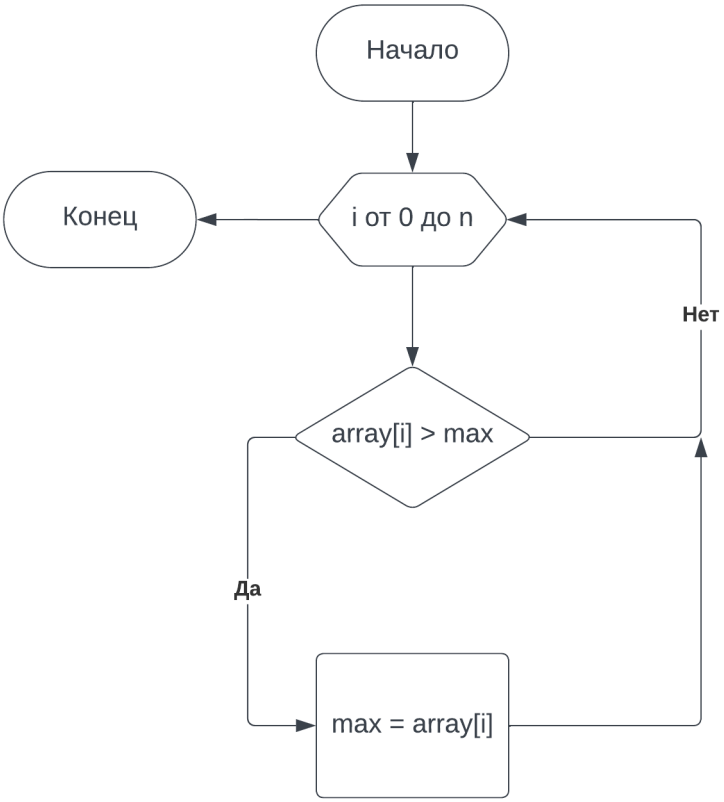
1. Описание рабочей среды

- Модель процессора: Intel Core i3-10110U CPU @ 2.10GHz
- Число ядер: 2
- Число потоков: 4
- Архитектура: x86-64
- ОС: Linux, дистрибутив Ubuntu v20.04
- RAM объем: 2x8192 MB
- RAM тип: DDR4
- Среда разработки: Visual Studio Code
- Компилятор: gcc v9.4.0
- Версия OpenMP: 201511

2. Анализ приведенного алгоритма

В задании лабораторной работы приведена программа, осуществляющая поиск максимального элемента в массиве.

Блоксхема алгоритма



Часть кода, использующая распараллеливание

```
int find_max(int *array, long array_size) {
  int rank, size;
  int local_max;
  int global_max;

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  int ibeg = rank * (ARRAY_SIZE / size);
  int iend = (rank == size - 1) ?
        ARRAY_SIZE : (rank + 1) * (ARRAY_SIZE / size);
  local_max = array[ibeg];

  for (int i = ibeg + 1; i < iend; ++i) {
    if (array[i] > local_max) {
        local_max = array[i];
    }
  }

  MPI_Reduce(&local_max, &global_max, 1, MPI_INT,
                            MPI_MAX, 0, MPI_COMM_WORLD);
  return global_max;
}
```

Описание работы алгоритма

Функциями `MPI_Comm_size` и `MPI_Comm_rank` определяются колличество потоков и его номер соответственно. Зная эти величины, массив разделяется между каждым потоком. Найдя локальные максимумы, с помощью функции `MPI_Reduce` находится максимальный элемент, который возвращается из функции.

3. Анализ временных характеристик последовательного алгоритма

Описание эксперимента

- Измеряется время работы алгоритма на 100 различных массивах длиной 10 000 000 элементов. Находится среднее значение;

Экспериментальные показатели

- Среднее время работы алгоритма: 0.021431 [с];

4. Анализ временных характеристик параллельного алгоритма

Описание эксперимента

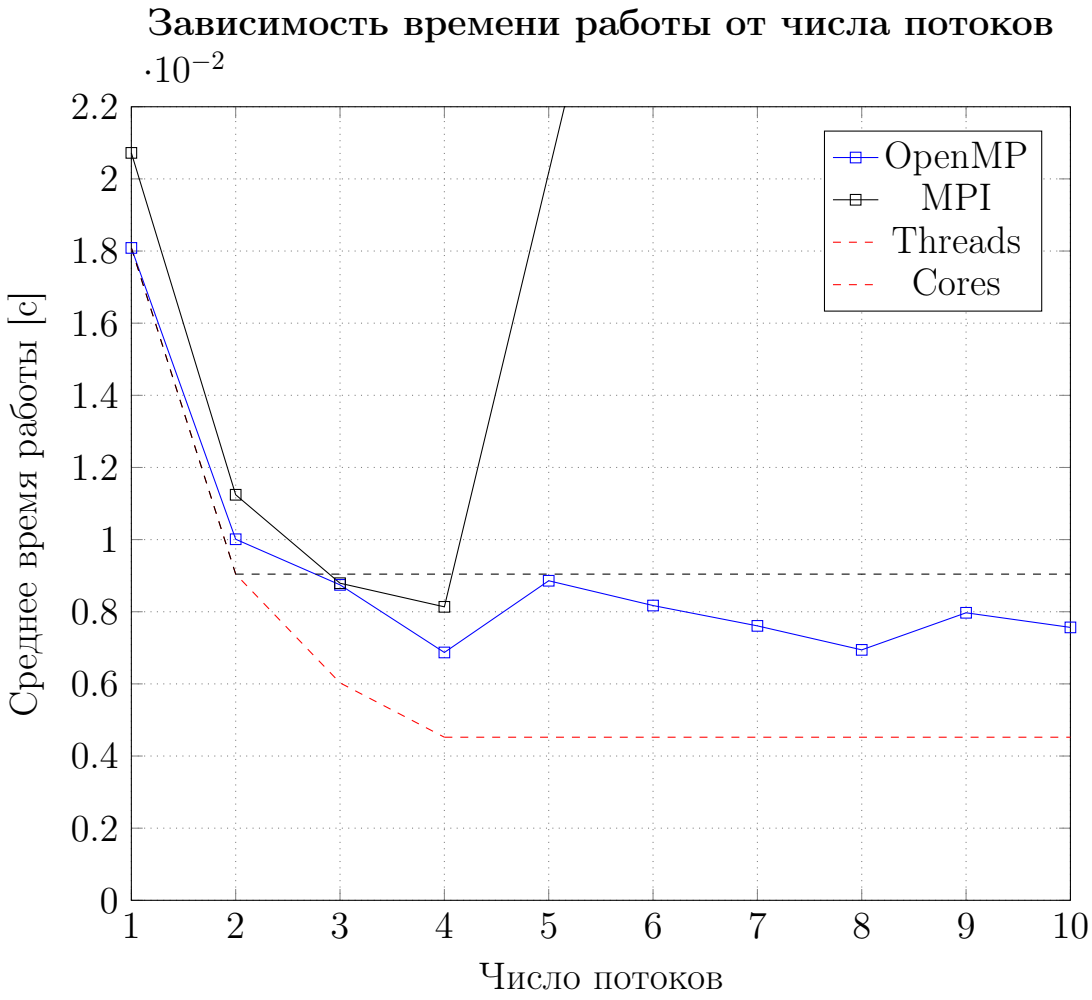
- измеряется время работы алгоритма для одного и того же массива, но на разном числе потоков: от 1 до 10;
- измерения производятся для 100 различных массивов, размер массива 10 000 000 элементов.

Результаты измерений

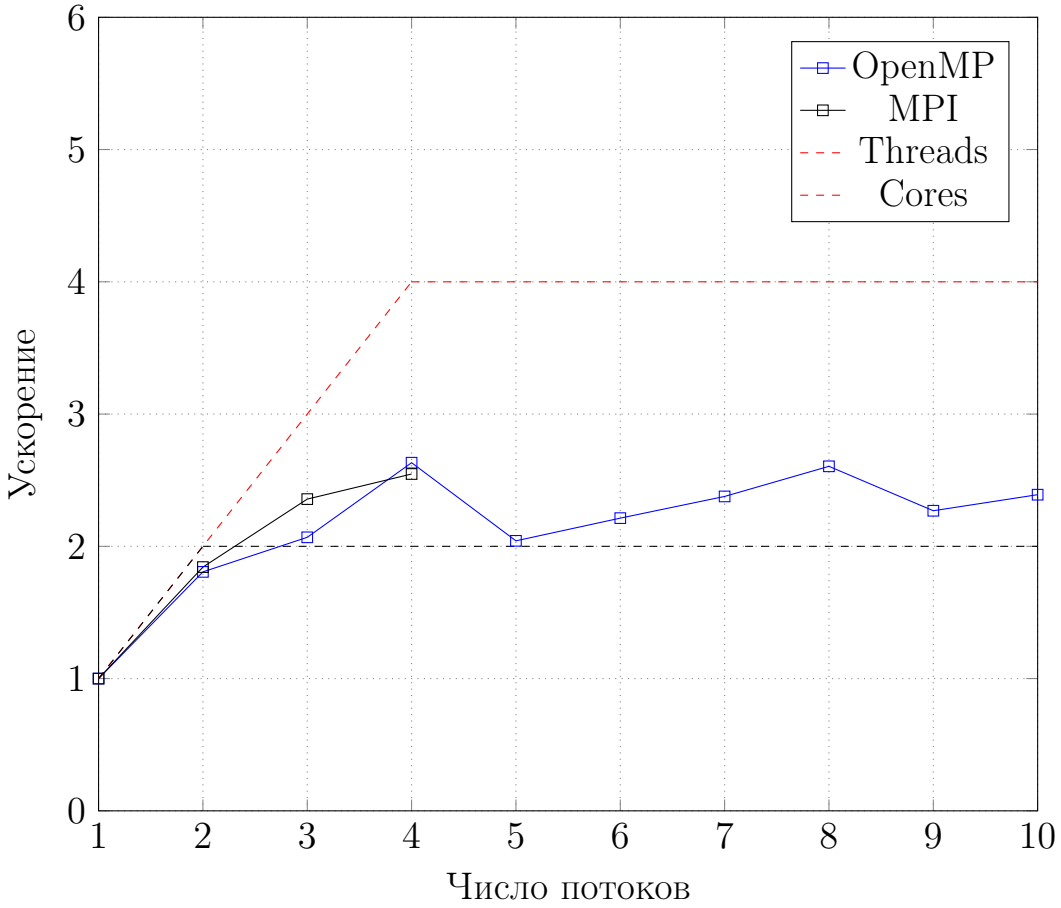
Следующая таблица содержит полученные в результате эксперимента данные: среднее время работы для различного числа потоков.

Число потоков	OpenMP	MPI
1	0.018086	0.020721
2	0.010010	0.011243
3	0.008745	0.008791
4	0.006871	0.008136
5	0.008859	-
6	0.008172	-
7	0.007608	-
8	0.006943	0.056306
9	0.007971	0.047310
10	0.007566	0.042144

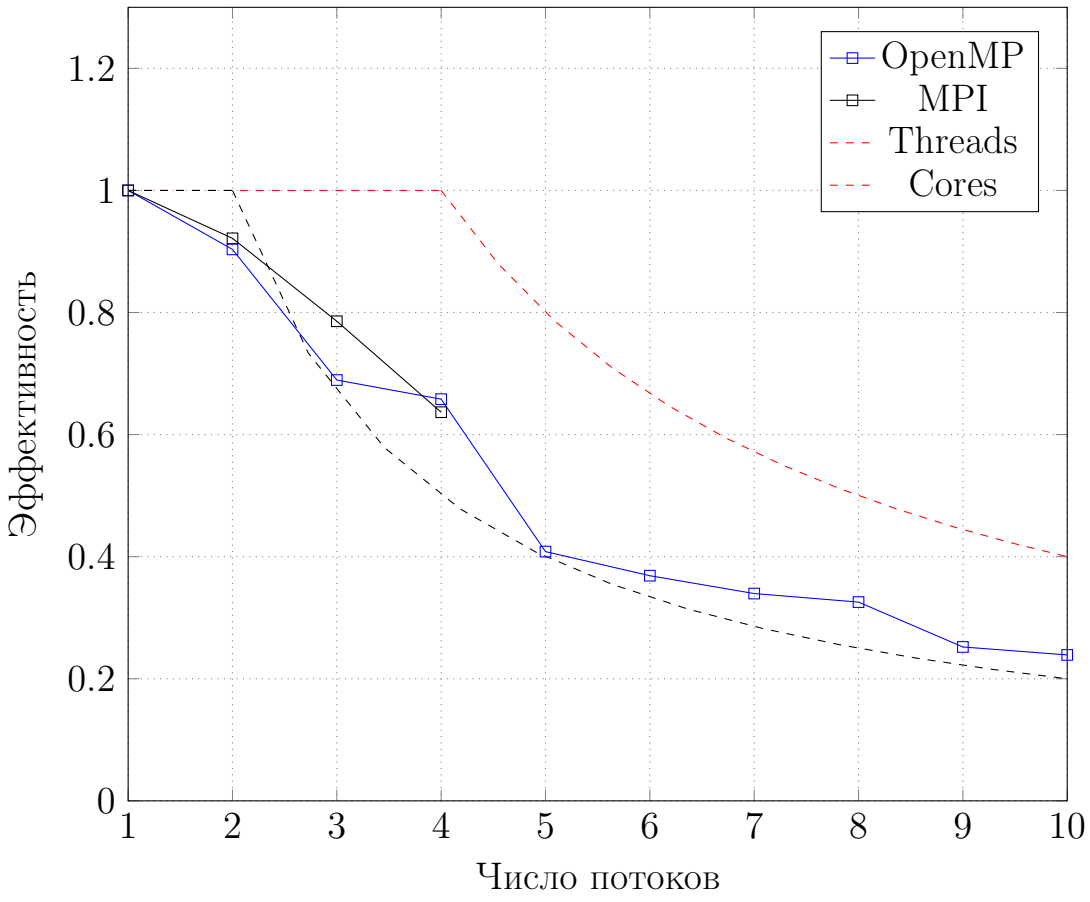
Графики


Зависимость времени работы от числа потоков
$\cdot 10^{-2}$
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10
Среднее время работы [c]
Число потоков
OpenMP
MPI
Threads
Cores

Зависимость ускорения от числа потоков



Зависимость эффективности от числа потоков



5. Заключение

В ходе лабораторной работы было измерено время работы алгоритма поиска максимального элемента в массиве для различного числа потоков используя MPI. По полученным данным были вычислены значения ускорения и эффективности. Построены соответствующие графики. Сравнены со значениями, полученными при использовании OpenMP.
Анализируя результаты эксперимента, можно обратить внимание на следующее:

- MPI для любого числа потоков показывает результат времени работы больше, чем OpenMP;
- эффективность и ускорение при использовании MPI выше на 2-х и 3-х потоках;
- наибольшая эффективность получена при 2-х потоках, что согласуется со средой.

6. Приложение

Код программы расположен на github
Запуск программы: `./run`