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

ОТЧЁТ

Лабораторная работа №1:
«Введение в параллельные вычисления. Технология OpenMP»

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

Москва 2023

Оглавление

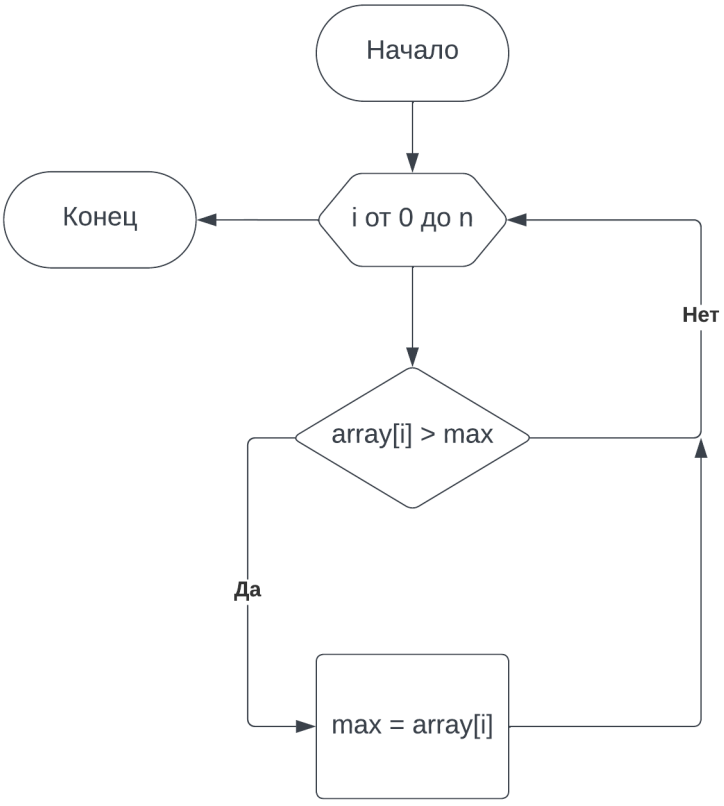
1. Описание рабочей среды

- Модель процессора:	Intel Core i3-10110U CPU @ 2.10GHz
- Число ядер:	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



Описание используемых директив OpenMP

`parallel` - Определяет параллельную область, которая представляет собой код, который будет выполняться несколькими потоками параллельно. Директива `parallel` была объявлена со следующими атрибутами:

- `num_threads()` - задаёт количество потоков в параллельном блоке (по умолчанию `parallel` использует все потоки);
- `reduction()` - определяет переменную, являющихся приватными для каждого потока, при выходе которые подвергаются операции редукции;
- `shared()` - объявляет, что переменные должны быть общими между всеми потоками;
- `default()` - задаёт стандартное поведение при встрече неинициализированной внутри потока переменной.

`for` - Разделяет работу цикла между потоками (без неё каждый поток обрабатывал бы весь массив).

Действие директивы `parallel` распространяется на следующий блок программного кода:

```
{
#pragma omp for
for(int i=0; i<count; i++) {
    if(array[i] > max) { max = array[i]; };
}
}
```

В свою очередь директива `for` действует на следующую строку:

```
for(int i=0; i<count; i++)
```

Описание работы алгоритма

Дерективой `parallel` объявляется блок кода, который будет исполняться параллельно. Далее внутри `for` потоки распередяют меду собой итерации цикла. Каждый поток ищет максимальный элемент своей части массива. Вычислив локальные максимумы, атрибут `reduction()` функцией `max` находит максимальный элемент из всех полученных с потоков.

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

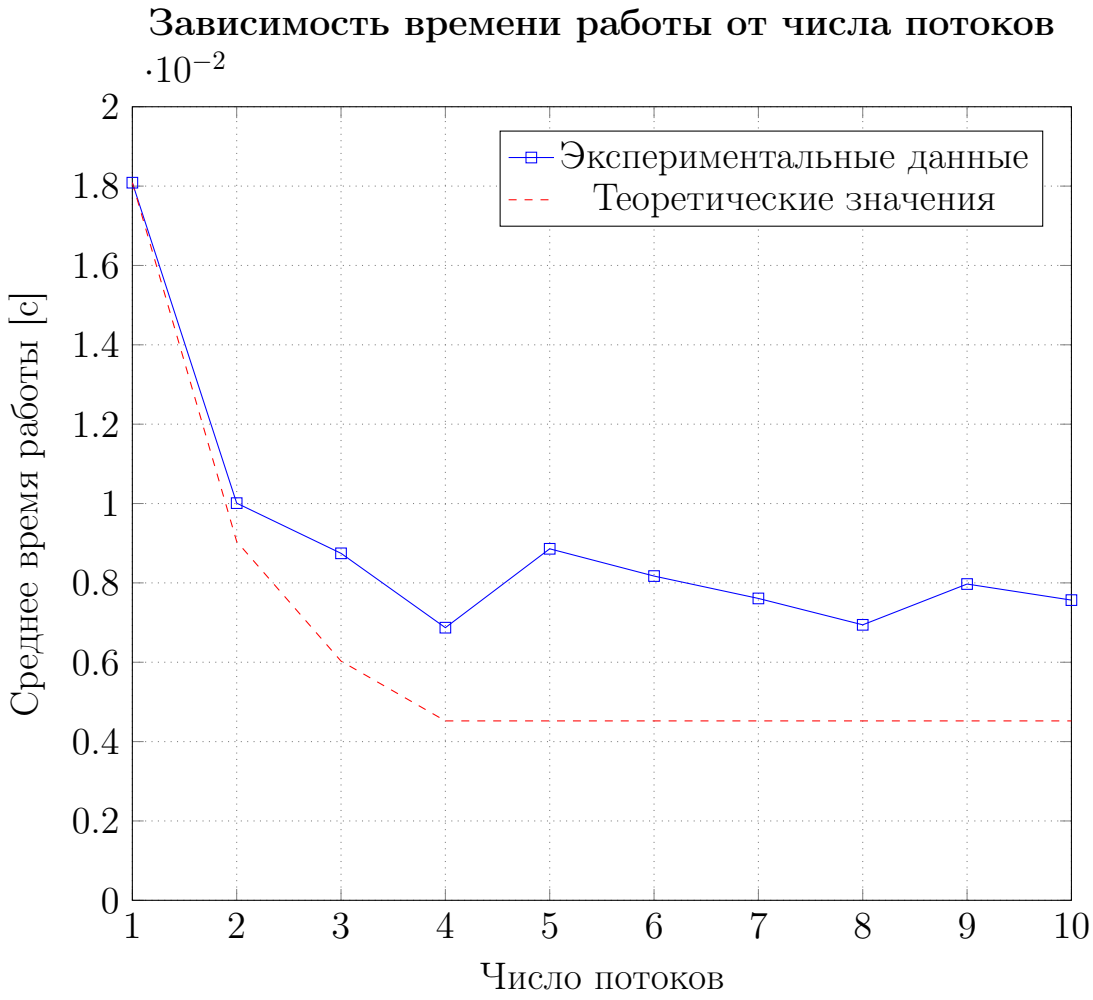
- измеряется время работы алгоритма для одного и того же массива, но на разном числе потоков: от 1 до 10;
- измерения производятся для 100 различных массивов, размер массива 10 000 000 элементов.

Результаты измерений

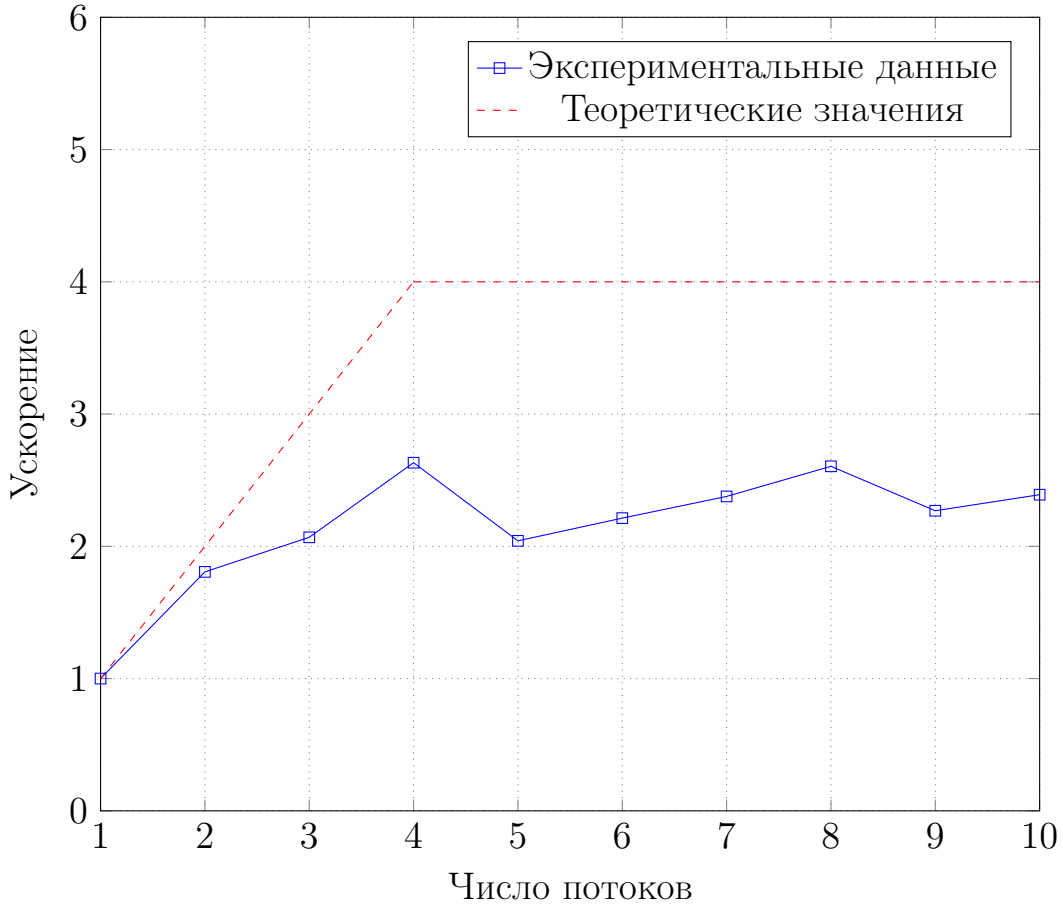
Следующая таблица содержит полученные в результате эксперимента данные: среднее время работы для различного числа потоков.

Число потоков	Среднее время работы
1	0.018086
2	0.010010
3	0.008745
4	0.006871
5	0.008859
6	0.008172
7	0.007608
8	0.006943
9	0.007971
10	0.007566

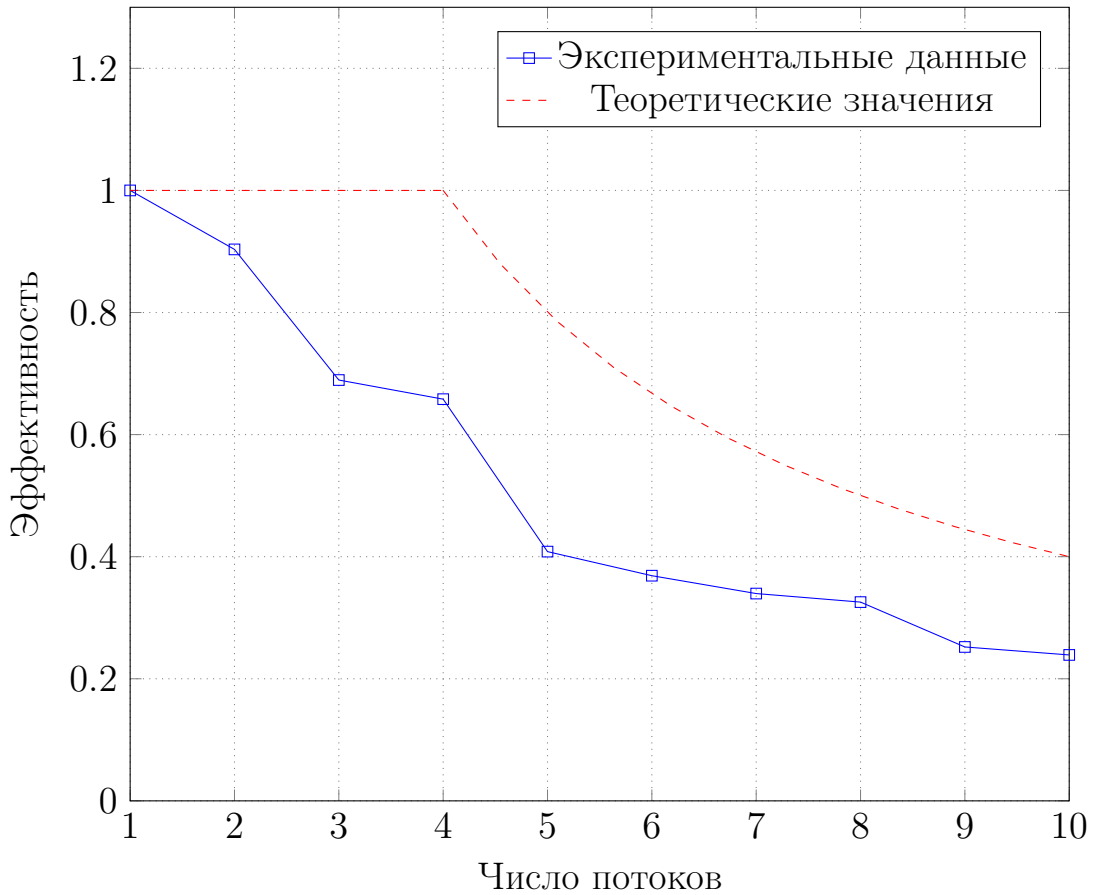
Графики


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

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. По полученным данным были вычислены значения ускорения и эффективности. Построены соответствующие графики.
Анализируя результаты эксперимента, можно обратить внимание на следующее:

- среднее время работы алгоритма уменьшается с ростом числа потоков до 4 включительно. Затем среднее время работы увеличивается при 5 потоках. При последующем увеличении числа потоков среднее время работы уменьшается;
- минимальное время работы алгоритма составляет 0.006871с на 4 потоках;
- максимальное ускорение составляет 2.632222 на 4 потоках.

Динамика значения среднего времени работы до 4 потоков близко к теоретическим значениям. Отклонение от теоретического графика связано с потреблением временных ресурсов при создании новых потоков, а также при синхронизации на выходе из параллельного блока.
Небольшой скачок при 5 потоках объясняется превышением требуемым числом потоков числа логических ядер.

6. Приложение

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