МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
КАФЕДРА №42 «КРИПТОЛОГИЯ И КИБЕРБЕЗОПАСНОСТЬ»

ОТЧЁТ

по дисциплине «Параллельное программирование»
Лабораторная работа №2
«Выделение ресурса параллелизма.Технология OpenMP»

Группа Б21-525

Студент Г.О. Шулындин

Преподаватель М.А. Куприяшин

Москва 2023

Оглавление

1. Описание рабочей среды

- Модель процессора: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx
- Число ядер: 8
- Архитектура: x86-64
- ОС: Linux, дистрибутив Ubuntu v22.04
- RAM объем: 2x4096 MB
- RAM тип: DDR4
- Используемая среда разработки: Visual Studio Code
- Компилятор: gcc v11.4.0
- Поддерживаемая версия OpenMP: 201511

2. Анализ приведенного линейного алгоритма

В задании приведен алгоритм, осуществляющий поиск заданного элемента в массиве.

Принцип работы алгоритма

Последовательно рассматриваем каждый элемент в массиве. Если рассматриваемый элемент равен искомому (`target`), то сохраняем индекс данного элемента, прерываем цикл.

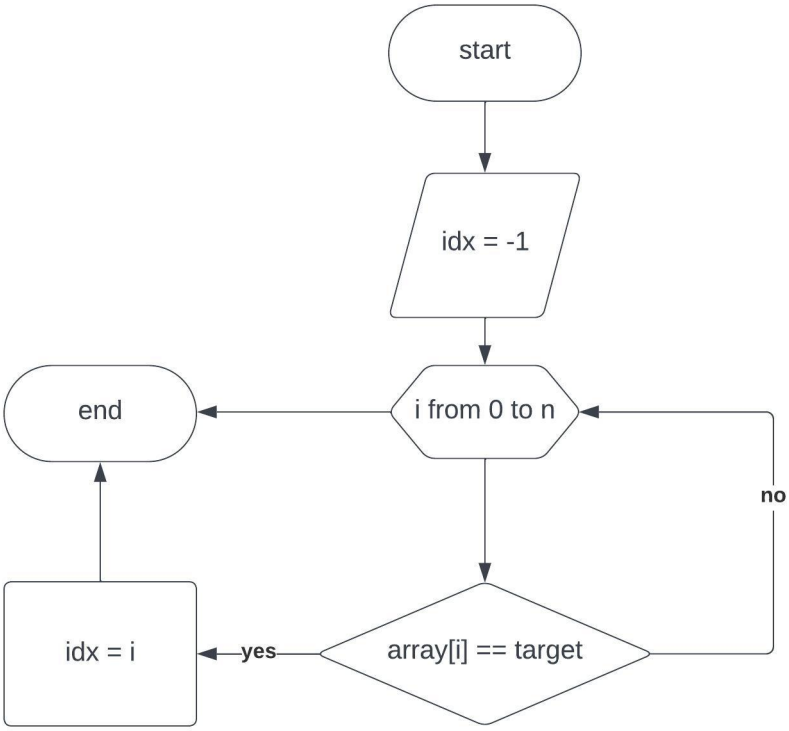
```
int idx = -1;
for(int i=0; i<array_length; i++) {
    if (array[i] == target) {
        idx = i;
        break;
    }
}
```

Временная сложность алгоритма

- В лучшем случае, искомый элемент является первым в массиве либо все элементы в массиве совпадают с искомым. Тогда в цикле `for` будет совершена лишь одна итерация, после чего цикл будет прерван. Временная сложность в этом случае составляет $O(1)$;
- В худшем случае, искомый элемент является последним в массиве либо в массиве нет исходного элемента. Тогда в цикле будет совершено n итераций, где n - длина массива. Временная сложность в этом случае составляет $O(n)$;
- В среднем случае, будем считать, что положение искомого элемента равновероятно относительно индекса массива (в том числе его может и вовсе не быть в массиве). Тогда среднее время работы составляет:

$$Mean = \frac{\sum_{n=0}^{n} O(i)}{n+1} = \frac{O((n+1) \cdot n/2)}{n+1} = O(n)$$

Блоксхема алгоритма



3. Построение параллельного алгоритма

Выявление области распараллеливания

В алгоритме линейного поиска заданного элемента в массиве имеет смысл распараллеливание цикла. В таком случае итерирование по циклу будет разделено между потоками, и время работы в среднем случае будет составлять $O(n/p)$, где p - число потоков.

Реализация параллельного алгоритма

```
1  double find_occurance_n_threads(int *array, int target, int
       array_length, int num_threads, int *idx) {
2
3      *idx = -1;
4      ....
5  #pragma omp parallel num_threads(num_threads) shared(array, target,
       array_length, idx)
6      {
7  #pragma omp for
8      for (int i = 0; i < array_length; i++) {
9          if (*idx == -1) {
10             if (array[i] == target) {
11 #pragma omp critical
12                 *idx = i;
13             }
14         } else { i = array_length; }
15     }
16     }
17     ....
18 }
```

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;
- `shared()` - объявляет, что одна или несколько переменных должны быть общими между всеми потоками;

`for` - приводит к разделу работы, выполняемой в цикле for внутри параллельной области, между потоками;

`critical` - определяет секцию кода, которая должна исполняться одним потоком единовременно.

4. Анализ временных характеристик параллельного алгоритма

Описание эксперимента

Условия эксперимента:

- измеряется время работы алгоритма для одного и того же массива на разном числе потоков: от 1 до 20;
- измерения производятся для 10 различных массивов одного типа;
- размер каждого массива 10 000 000 элементов.

Были выделены следующие типы массивов:

- искомый элемент является первым в массиве (Type 0);
- искомый элемент находится в начале массива (Type 1);
- искомый элемент находится в конце массива (Type 2);
- искомый элемент является последним в массиве (Type 3);
- искомый элемент находится в области середины массива (Type 4);
- искомого элемента нет в массиве (Type 5);
- все элементы в массиве соответствуют искомому (Type 6).

Результаты измерений

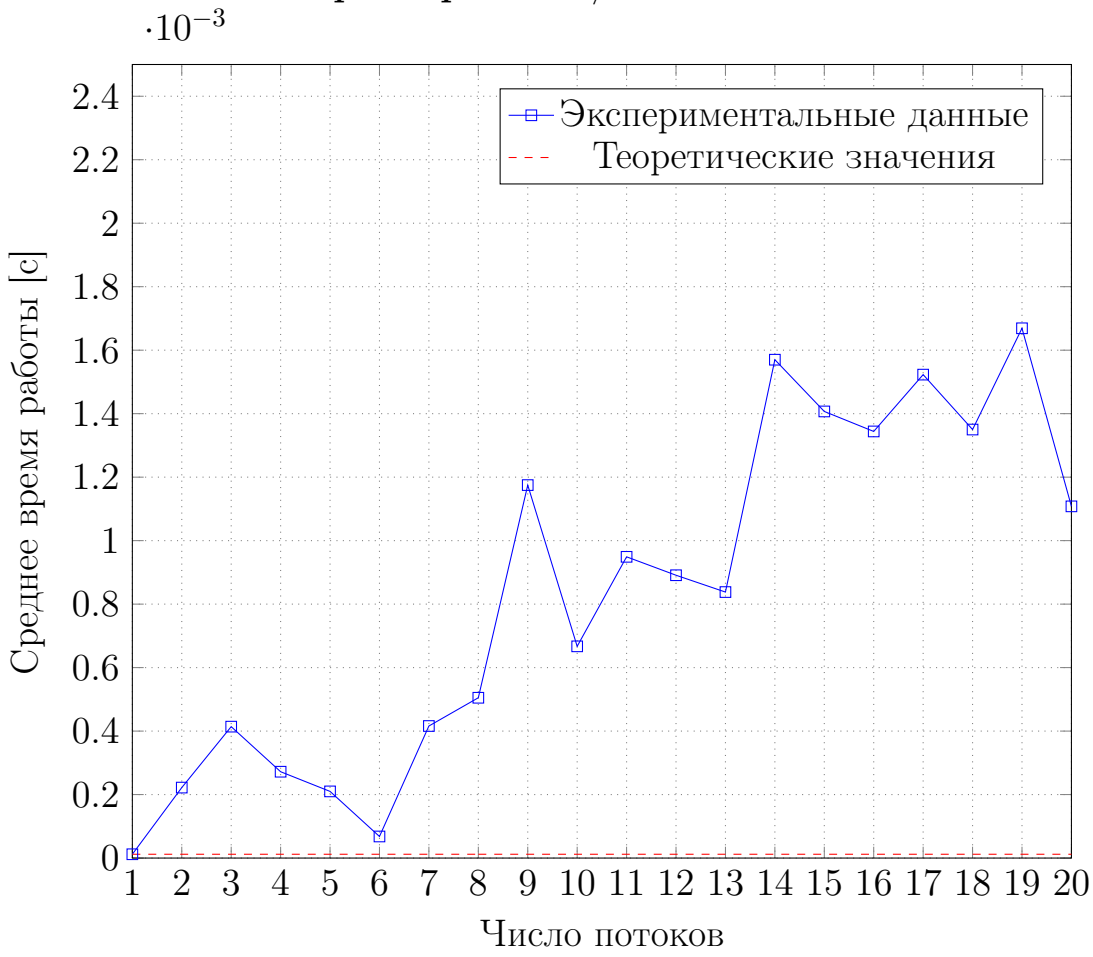
Следующая таблица содержит полученные в результате эксперимента данные: среднее время работы на различном числе потоков для массивов различных типов.

Thds num.	Type 0	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6
1	1.2e-05	9.9e-05	0.03461	0.034661	0.017943	0.03501	1.2e-05
2	0.000222	0.000242	0.02544	0.024817	0.001836	0.02209	0.000246
3	0.000414	0.00042	0.020445	0.016774	0.01171	0.020819	0.000592
4	0.000272	0.000512	0.013801	0.01427	0.001723	0.015196	0.000301
5	0.00021	0.000287	0.009396	0.008757	0.008006	0.012434	0.000185
6	6.8e-05	0.000178	0.008857	0.008549	0.001227	0.01094	5.1e-05
7	0.000416	0.000179	0.007292	0.007816	0.005397	0.009601	0.00024
8	0.000505	0.000238	0.006326	0.006554	0.001142	0.008261	0.000305
9	0.001175	0.000243	0.00808	0.007656	0.003475	0.008572	0.000184
10	0.000667	0.000285	0.004305	0.005004	0.001773	0.008189	0.000167
11	0.000949	0.000647	0.005867	0.004616	0.003339	0.007684	0.000182
12	0.000891	0.00095	0.006106	0.005478	0.002499	0.00772	0.000204
13	0.000838	0.000342	0.005704	0.005016	0.002136	0.006913	0.000214
14	0.00157	0.000833	0.004449	0.004786	0.001975	0.006576	0.000215
15	0.001407	0.000555	0.005035	0.005479	0.00317	0.006436	0.000203
16	0.001344	0.000744	0.005358	0.004825	0.002401	0.005828	0.000201
17	0.001523	0.000527	0.005911	0.005333	0.002519	0.006948	0.000194
18	0.00135	0.001148	0.005733	0.005194	0.002052	0.006773	0.000176
19	0.001669	0.001359	0.00589	0.005255	0.002354	0.006475	0.000158
20	0.001108	0.000903	0.005681	0.005893	0.002694	0.006315	0.000176

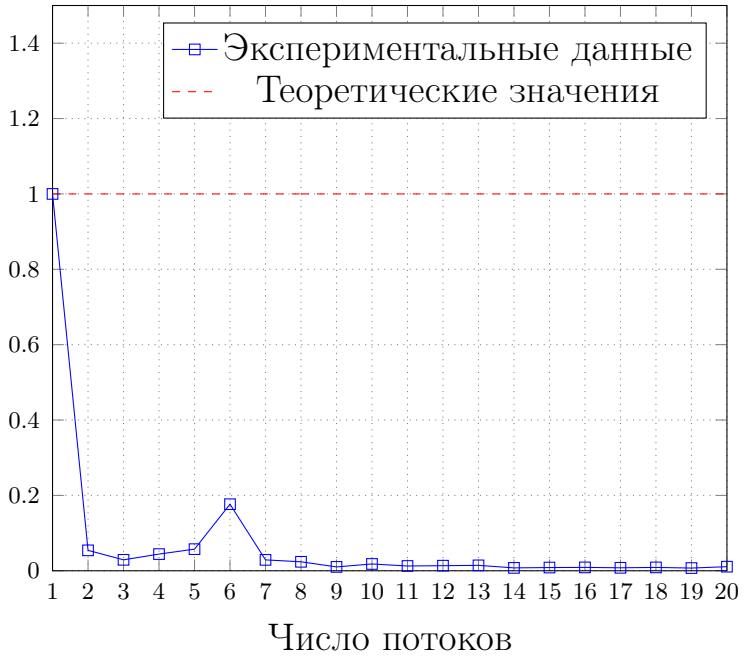
Графики

Искомый элемент является первым в массиве (Type 0)

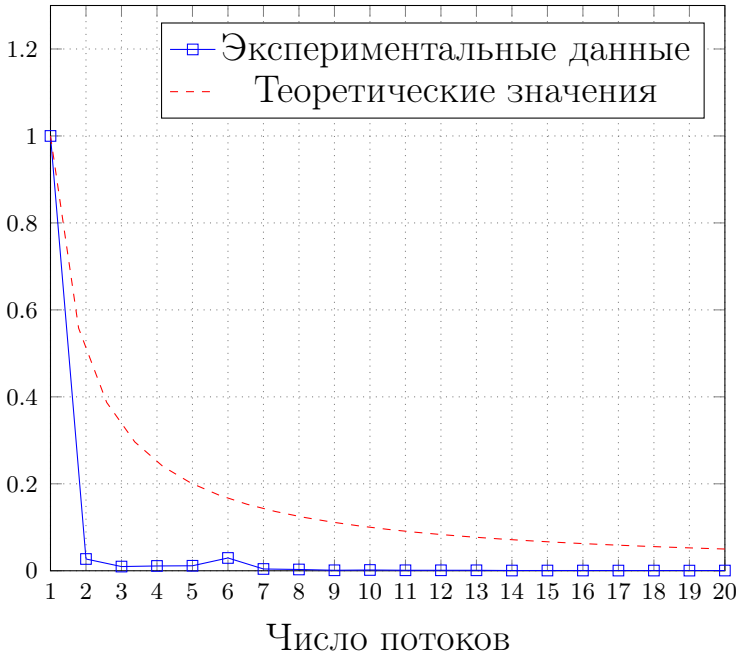
Время работы / число потоков



Ускорение / число потоков

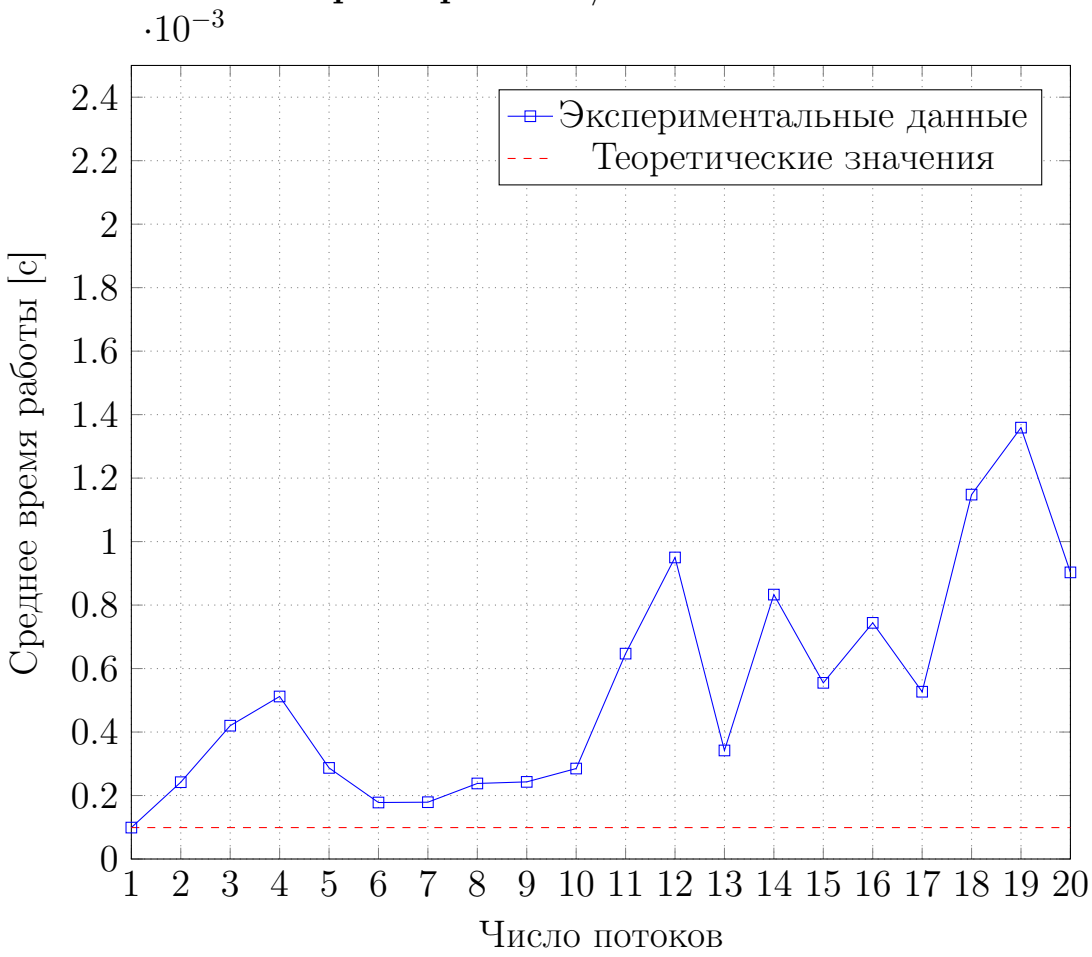


Эффективность / число потоков

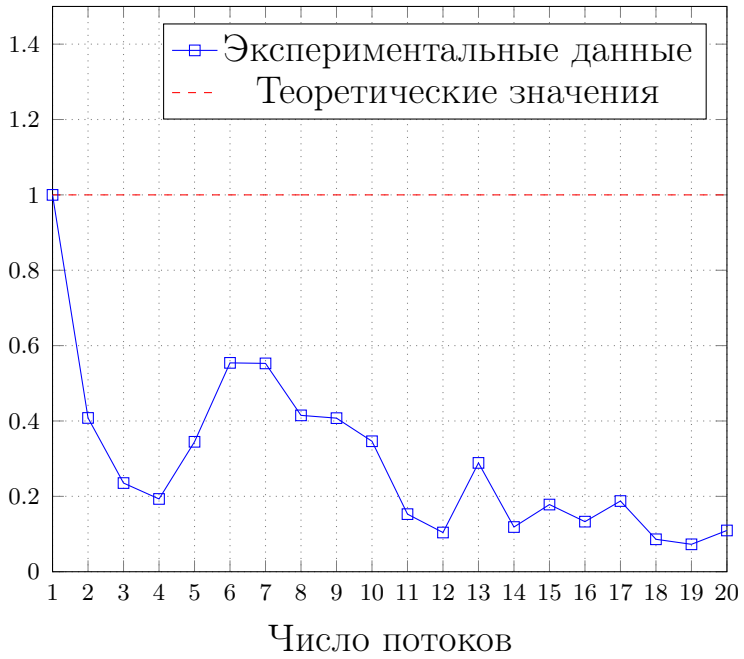


Искомый элемент находится в начале массива (Type 1)

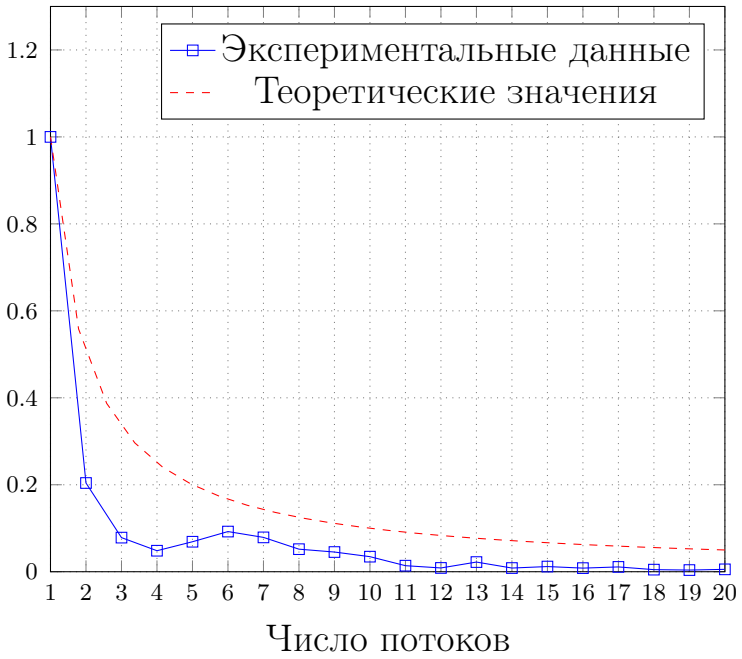
Время работы / число потоков



Ускорение / число потоков

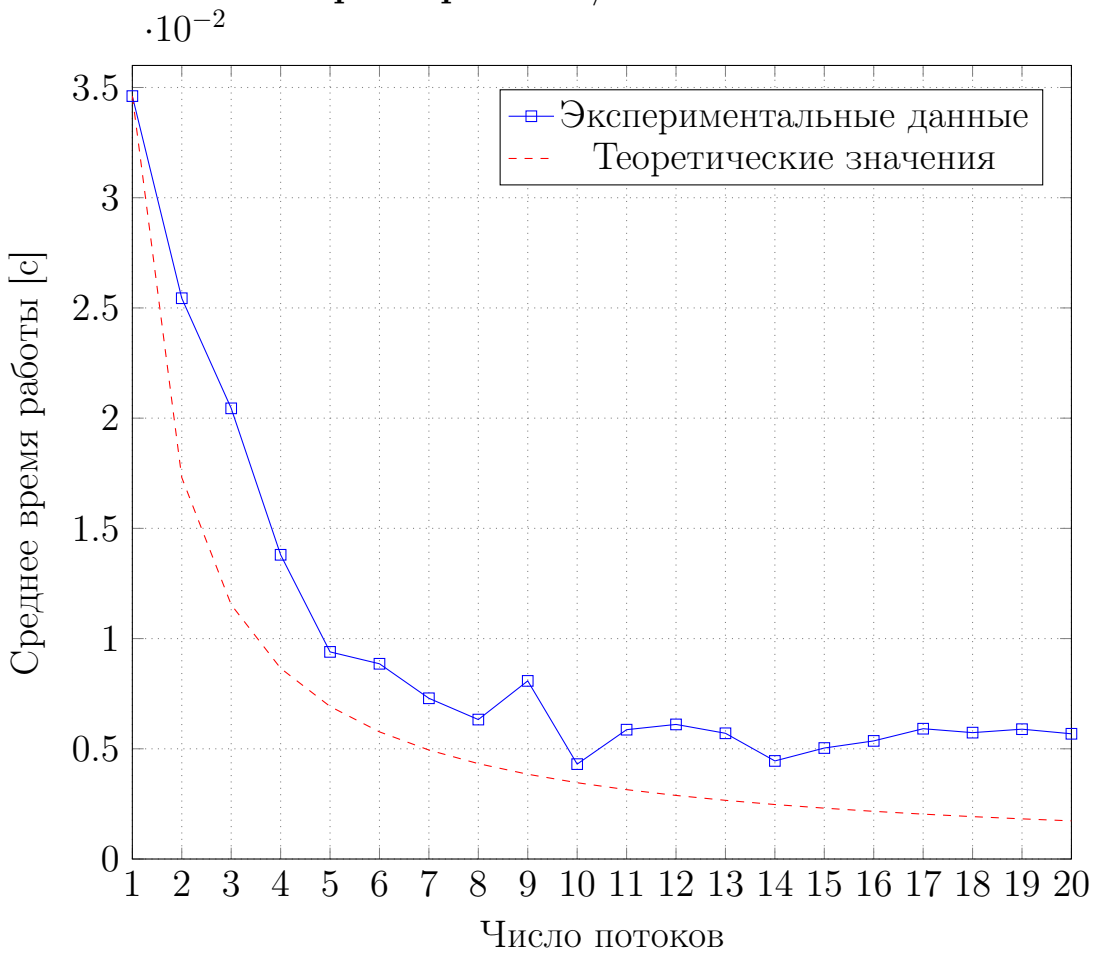


Эффективность / число потоков

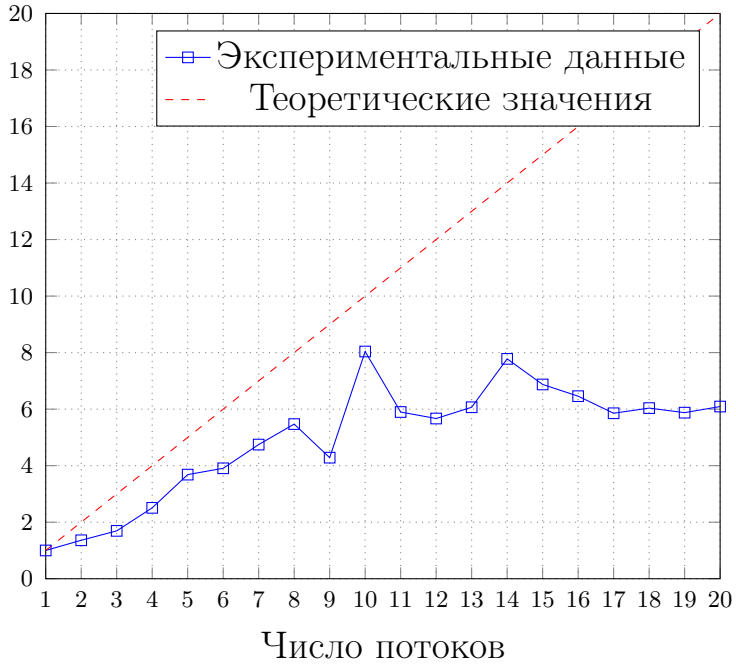


Искомый элемент находится в конце массива (Type 2)

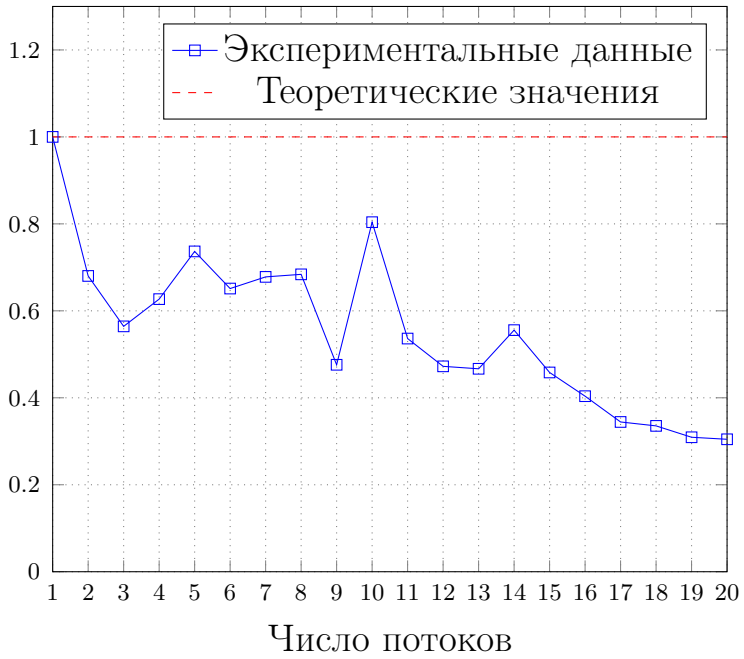
Время работы / число потоков



Ускорение / число потоков

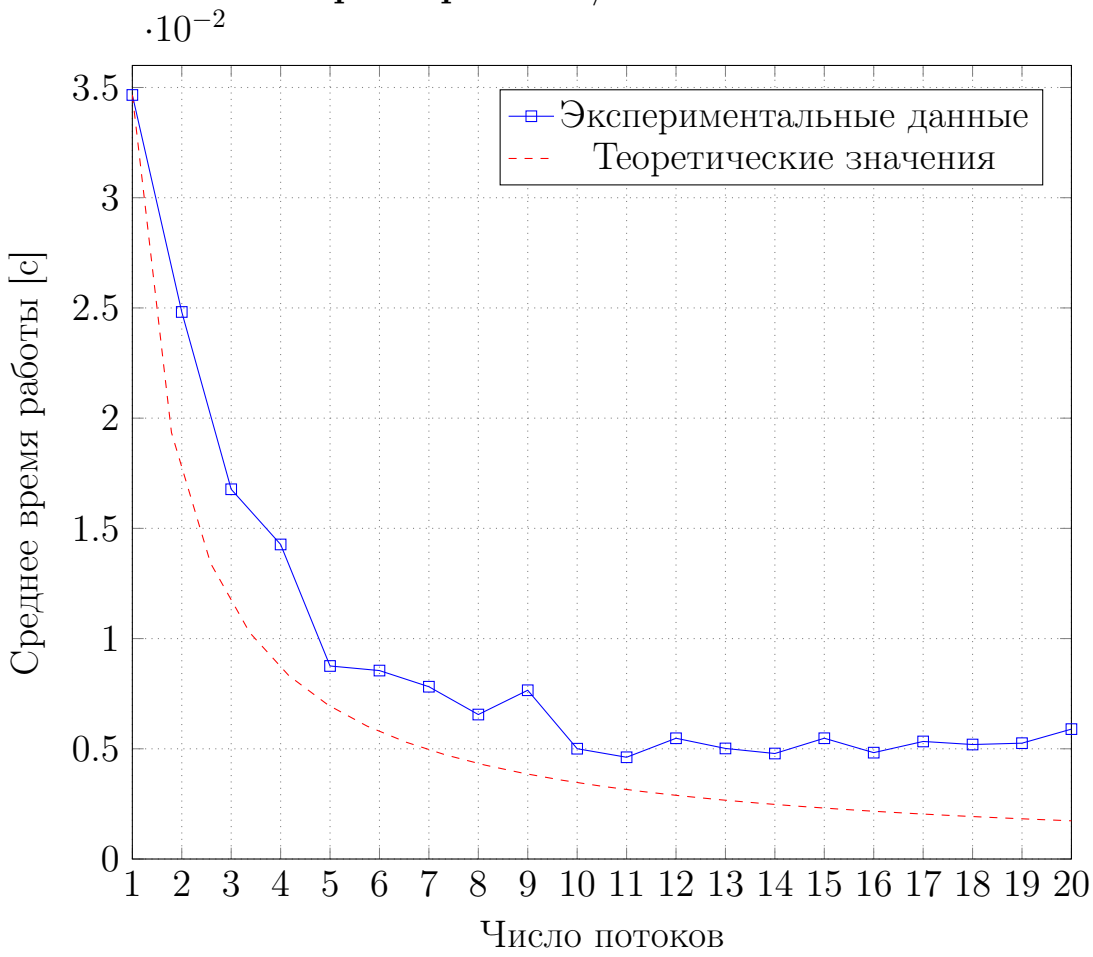


Эффективность / число потоков

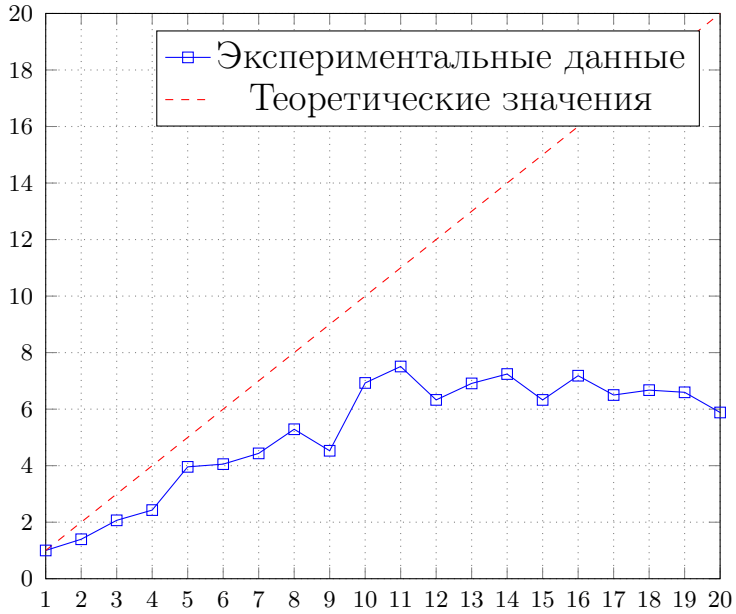


Искомый элемент является последним в массиве (Type 3)

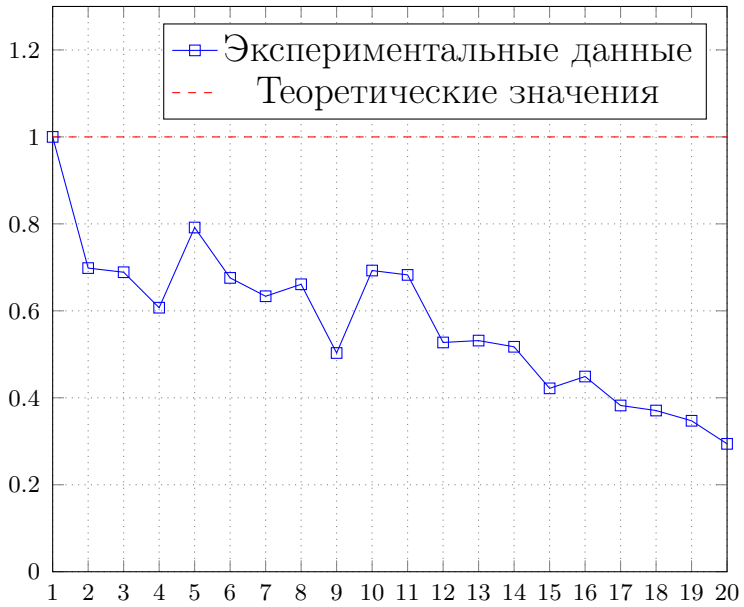
Время работы / число потоков



Ускорение / число потоков

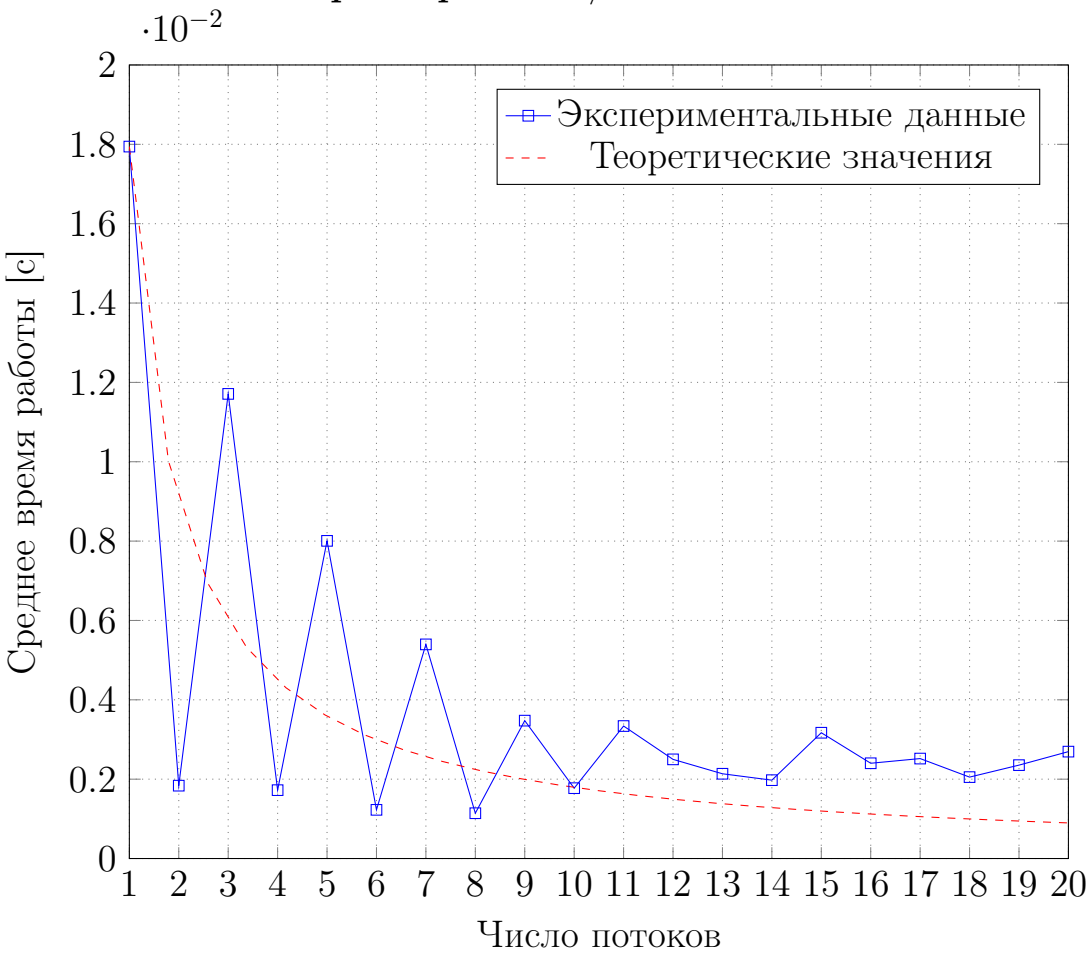


Эффективность / число потоков

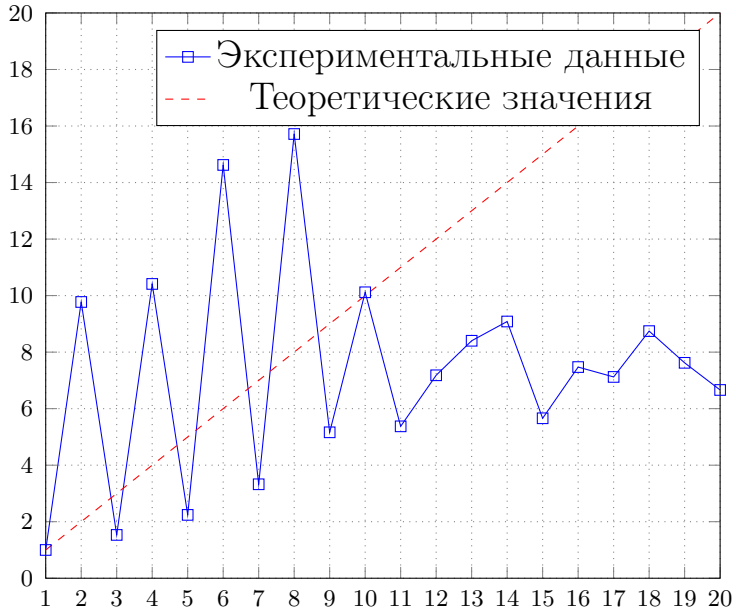


Искомый элемент находится в области середины массива (Type 4)

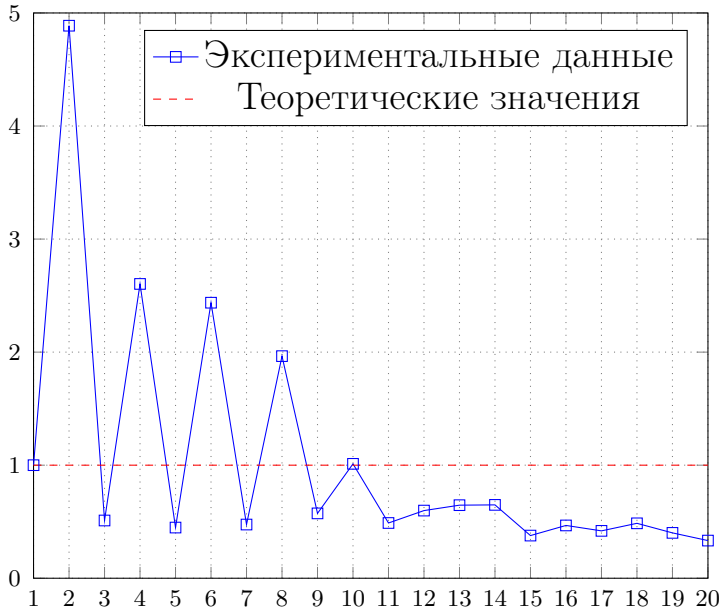
Время работы / число потоков



Ускорение / число потоков

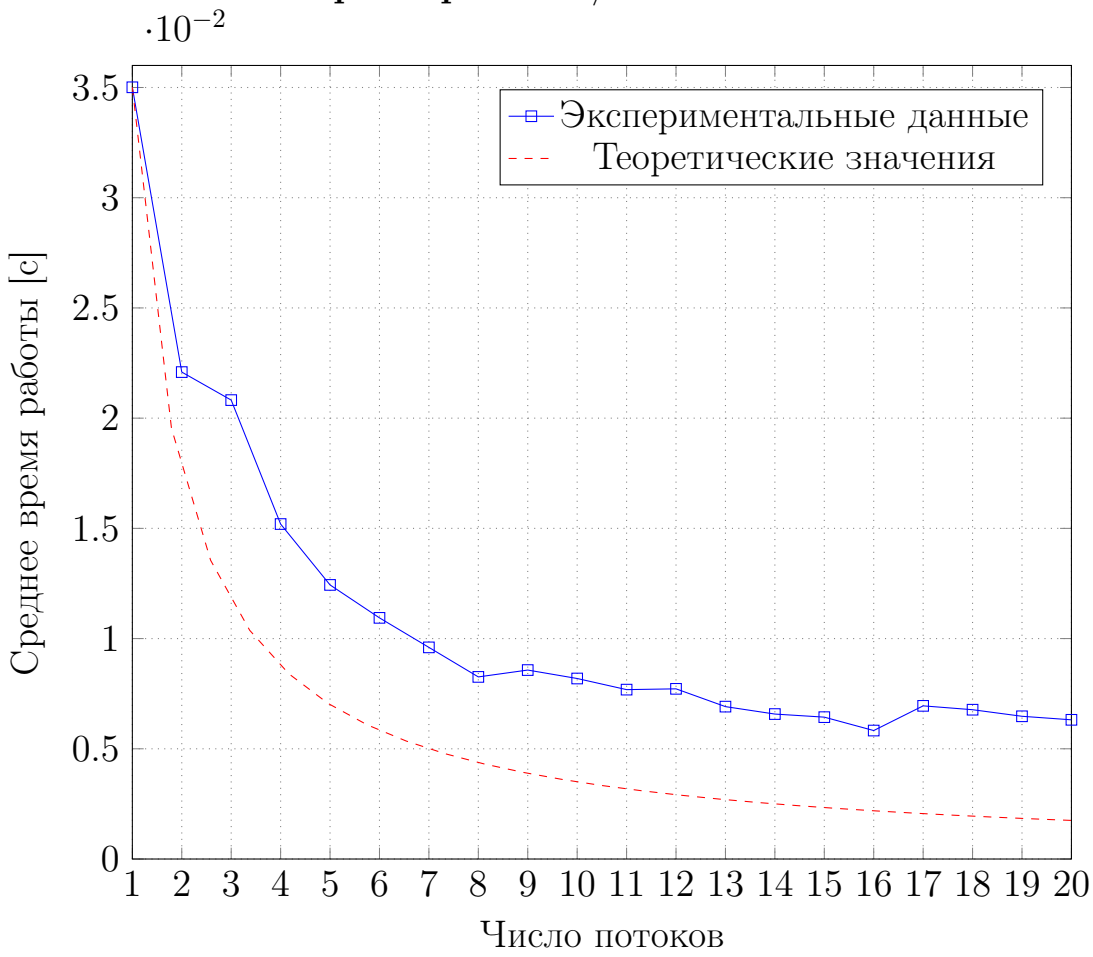


Эффективность / число потоков

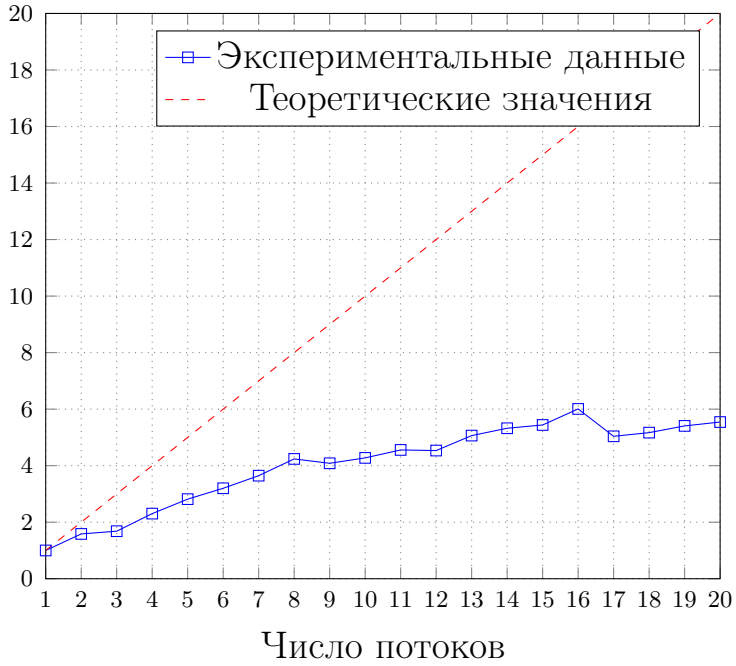


Искомого элемента нет в массиве (Type 5)

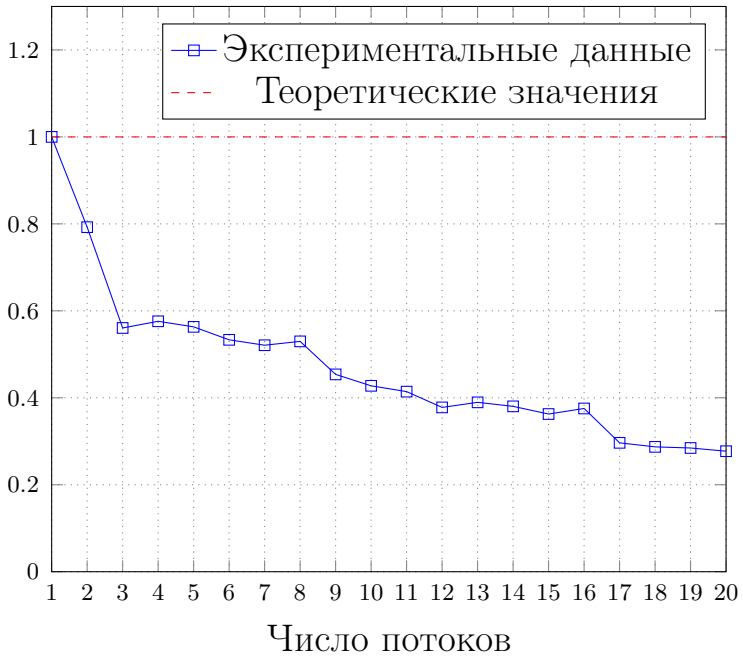
Время работы / число потоков



Ускорение / число потоков

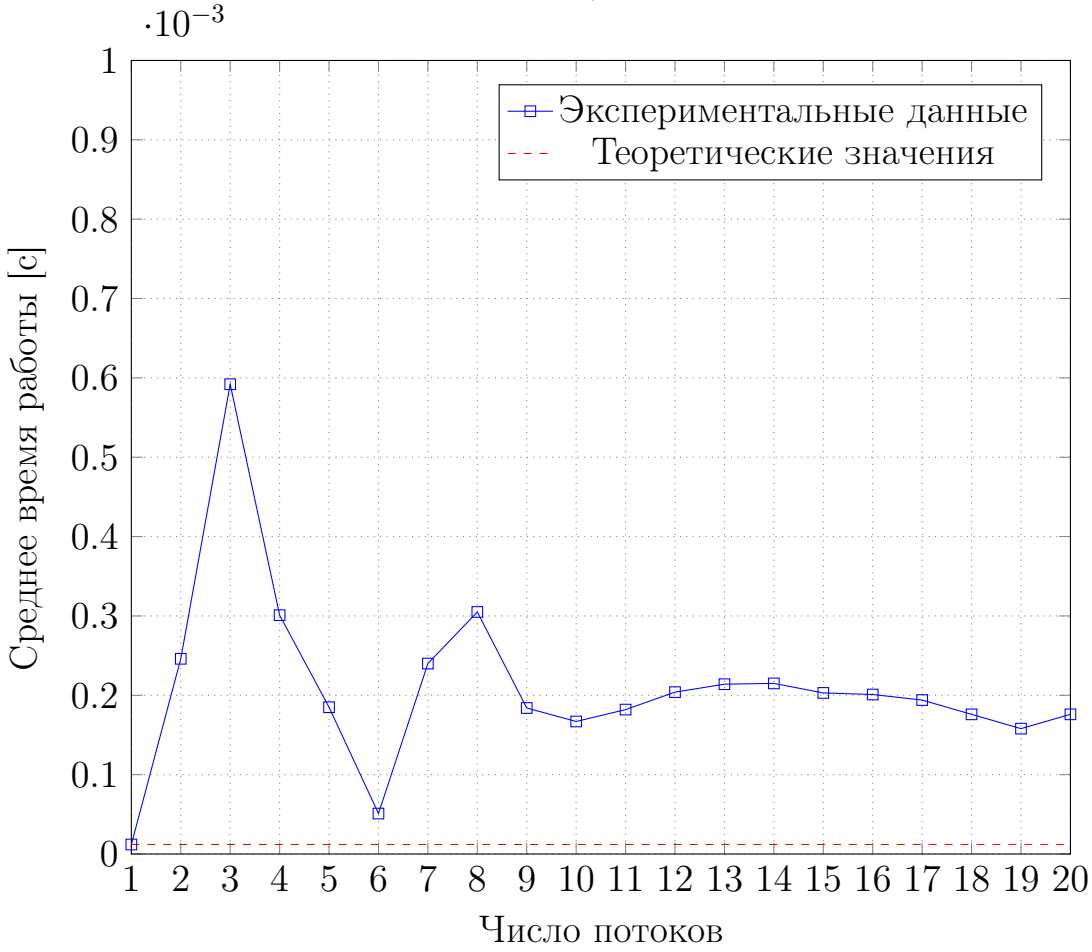


Эффективность / число потоков

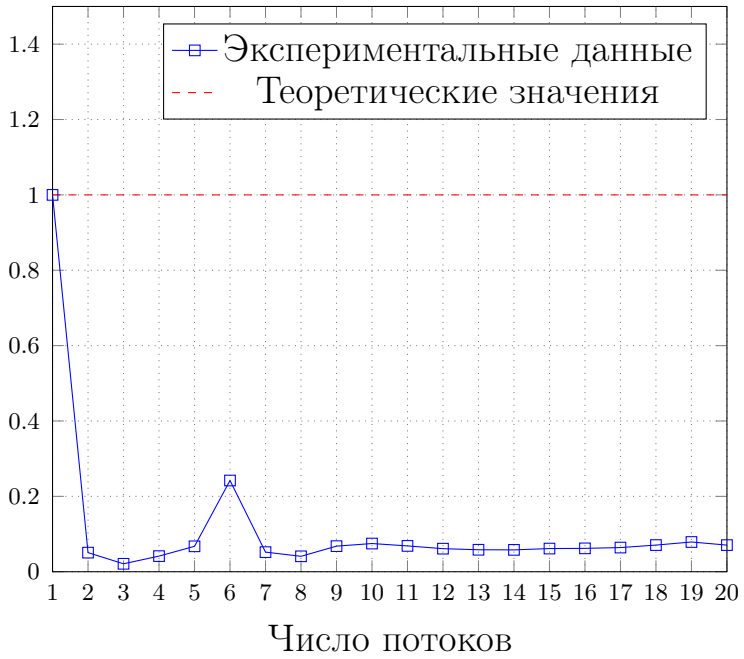


Все элементы в массиве соответствуют искомом (Type 6)

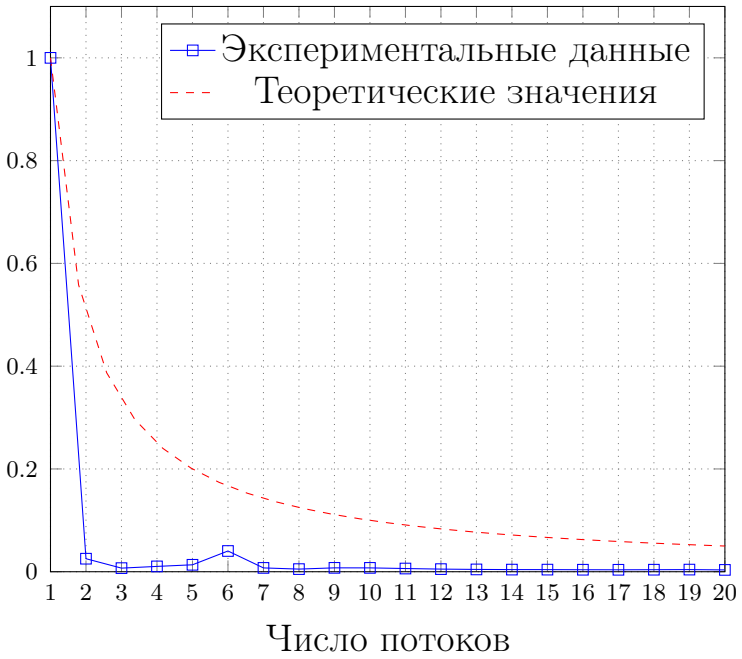
Время работы / число потоков



Ускорение / число потоков



Эффективность / число потоков



5. Заключение

В ходе лабораторной работы было измерено среднее время работы алгоритма линейного поиска элемента в массиве для различного числа потоков и для различных типов массивов. По полученным данным были вычислены значения ускорения и эффективности. Построены соответствующие графики.

Анализируя результаты, можно сделать следующие выводы:

- Распараллеливание нецелесообразно, когда искомый элемент находится в начале массива либо когда все элементы в массиве соответсвуют искомому. В этом случае линейный алгоритм, как и параллельный выпалняется за $O(1)$, но при прочих равных параллельный алгоритм использует дополнительные ресурсы для распараллеливания. Соответсвенно, параллельный алгоритм работает на порядок медленнее линейного.
- Однако, параллельный алгоритм эффективен и целесообразен в остальных случаях. Так, если искомый элемент находится в конце массива, максимальное ускорение составляет 8.039008 на 9 потоках. Если элемент является последним в массиве - 7.508941 на 11 потоках. Если элемента нет в массиве - 6.006942 на 16 потоках.
- Особенно интересен случай, когда искомый элемент находится в области середины массива. На 2, 4, 6, 8 потоках алгоритм выполнялся быстрее теоретических значений. Это может объясняться тем, что искомый элемент находился в начале подмассива, отданного какому-то потоку, следовательно, элемент находился быстро и цикл завершался. Максимальное ускорение составляет 15.716852 на 8 потоках.

6. Приложение

Код программы, таблицы расположены на github

Запуск программы: `make all`