Лабораторная работа 1.

Остриков Денис Александрович
Б20-505
2022

Рабочая среда.

Версия OpenMP: 201511.

Алгоритм.

В переменной count - число элементов в одном массиве (10000000), arrays_size - число массивов для анализа (10), max = -1 изначально.

Изначально идет выделение памяти для массивов и заполнение рандомными числами с одним и тем же сидом.

1. Один поток.
   В каждом массиве в одном потоке сравниваются поочередно max и i-ый элемент и записываем в max наибольший из двух. После того как мы пройдем весь массив, получим максимальный элемент в max.
2. Многопоточность.
   В каждом массиве разбиваем цикл for на n потоков, которые, в свою очередь, постепенно увеличиваются в целях таймирования и установления, правда ли разделение на потоки даёт прирост к скорости.
   Директива

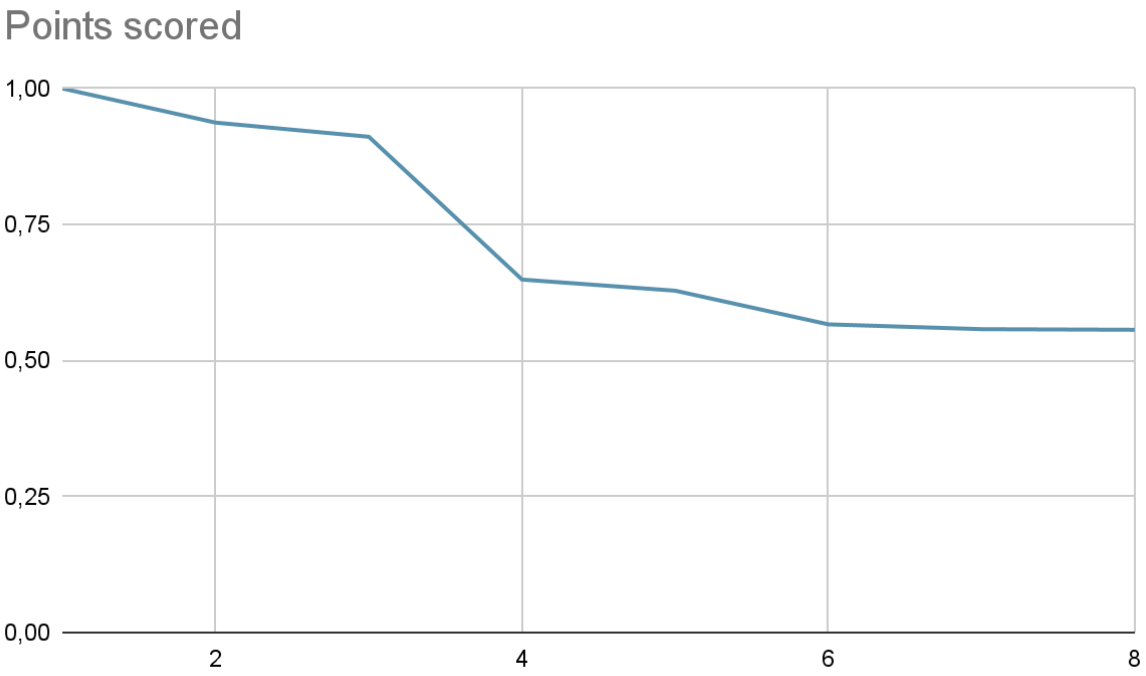
```
#pragma omp parallel num_threads(threads) shared(arrays, count, i) reduction(max: max) default(none)
```

   Задает параллельно исполняемый блок, выполняемый в threads потоках. В нем используются arrays, count, i общие переменные, то есть они не копируются для каждого отдельного потока, будь эти переменные не в shared a в private. Reduction задаёт редуктивную операцию, в данном случае редуктивная операция - max над переменной max, то есть после выполнения потока из всех переменных max мы получим наибольшее значение.

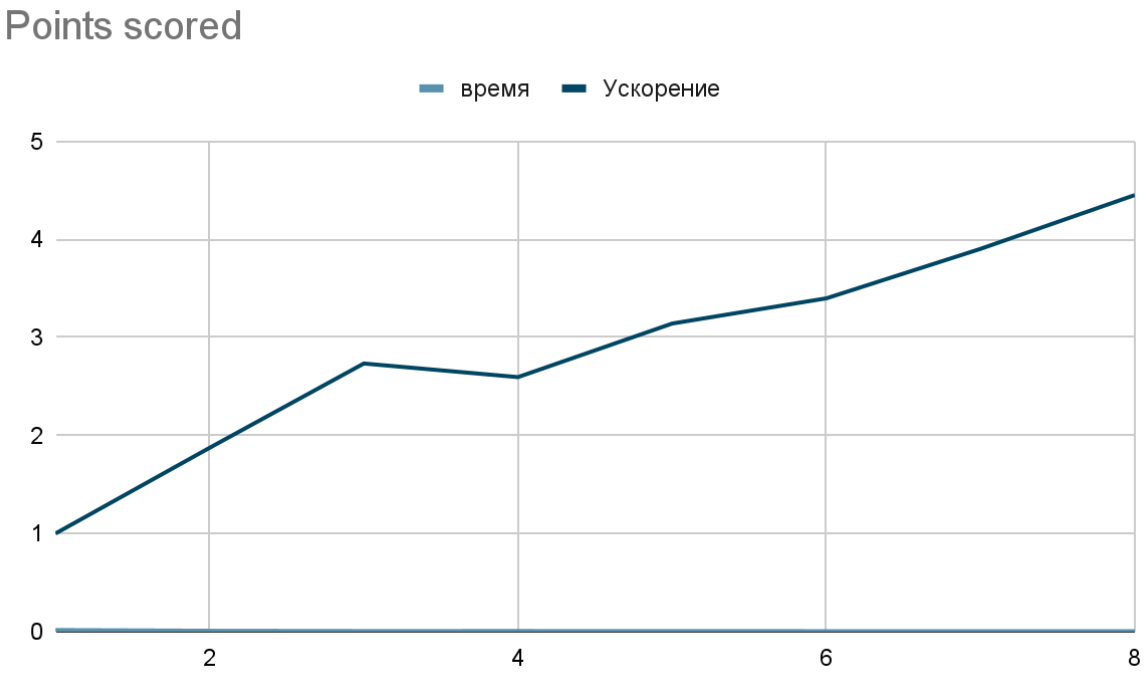
```
#pragma omp for
```

   Данная директива позволяет разбить цикл на кусочки, которые выполняются отдельно в потоках. Если бы её не было, каждый поток выполнил бы по count операций, тем самым не ускорив выполнение программы ни на грамм, только замедлил бы её разбиением на потоки.

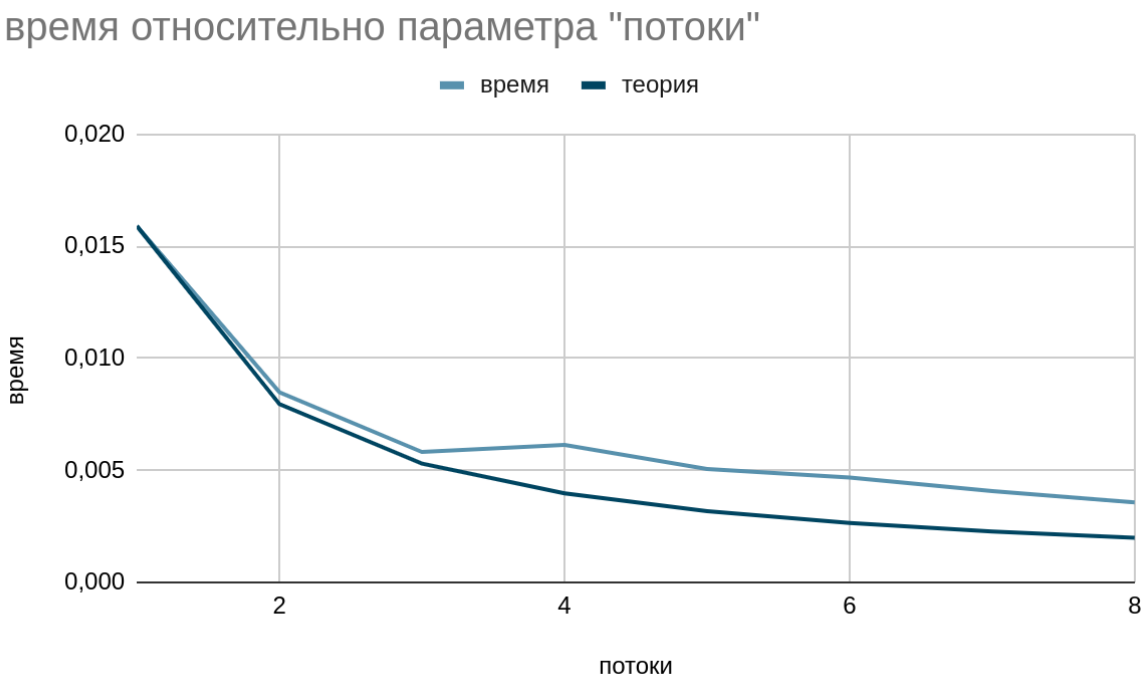
Таймирование:
Эффективность:



Ускорение:



Теоретическое и практическое время от количества потоков:



Заключение.

Было проведено измерение и анализ эффективности алгоритма. Полученные результаты не сильно отличаются от теоретических представлений. Заметно, что при больших значениях количества потоков, параллельное выполнение того же алгоритма значительно ускорят программу.