Отчет по лабораторной работе 6

Параметры виртуальной машины, на которой проводился эксперимент:

- Общий объем оперативной памяти: 4005120 kB;
- Объем раздела подкачки: 4004860 kB;
- Размер страницы виртуальной памяти: 4 kB;
- Процессор: по умолчанию в UTM с количеством ядер, указанном в эксперименте.

Параметры хостового компьютера, на котором проводился эксперимент:

- Общий объем оперативной памяти: 16777216 kB;
- Процессор: Apple M1, 8-ядерный процессор с 4 ядрами производительности и 4 ядрами эффективности.

Эксперимент №1

За сложный алгоритм было взято вычисление геометрического ряда Маклорена:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots + x^n,\ |x| < 1.$$

Время выполнения для любого входного параметра составляет 3 секунды при использовании 1 ядра. Реализацию можно посмотреть в файле **function.sh**.

Последовательный запуск программы с использованием 1 ядра

Реализация скрипта, выполняющего последовательно N вычислений, находится в файле **1_1b.sh**.
Реализация скрипта, выполняющего 10 запусков для N в диапазоне от 1 до 20, находится в файле **1_1c.sh**.
Данные записывались в файл **log_1_1**, после чего были посчитаны средние значения, которые отображены на Рис.1.

Параллельный запуск программы с использованием 1 ядра

Реализация скрипта, выполняющего параллельно N вычислений, находится в файле **1_2a.sh**.
Реализация скрипта, выполняющего 10 запусков для N в диапазоне от 1 до 20, находится в файле **1_2b.sh**.
Данные записывались в файл **log_1_2**, после чего были посчитаны средние значения, которые отображены на Рис.2.

Последовательный запуск программы с использованием 2 ядер

Необходимые скрипты уже реализованы.
Данные записывались в файл **log_1_3**, после чего были посчитаны средние значения, которые отображены на Рис.3.

Параллельный запуск программы с использованием 2 ядер

Необходимые скрипты уже реализованы.
Данные записывались в файл **log_1_4**, после чего были посчитаны средние значения, которые отображены на Рис.4.

Эксперимент №2

Для описанного алгоритма было реализовано 2 скрипта: **create_files.sh**, который создает необходимые файлы, и **write.sh**, который для каждого числа в файле записывает в конец удвоенное значение.

Последовательный запуск программы с использованием 1 ядра

Реализация скрипта, выполняющего последовательно N вычислений, находится в файле **2_1b.sh**.
Реализация скрипта, выполняющего 10 запусков для N в диапазоне от 1 до 20, находится в файле **2_1c.sh**.
Данные записывались в файл **log_2_1**, после чего были посчитаны средние значения, которые отображены на Рис.5.

Параллельный запуск программы с использованием 1 ядра

Реализация скрипта, выполняющего параллельно N вычислений, находится в файле **2_2a.sh**.
Реализация скрипта, выполняющего 10 запусков для N в диапазоне от 1 до 20, находится в файле **2_2b.sh**.
Данные записывались в файл **log_2_2**, после чего были посчитаны средние значения, которые отображены на Рис.6.

Последовательный запуск программы с использованием 2 ядер

Необходимые скрипты уже реализованы.
Данные записывались в файл **log_2_3**, после чего были посчитаны средние значения, которые отображены на Рис.7.

Параллельный запуск программы с использованием 2 ядер

Необходимые скрипты уже реализованы.
Данные записывались в файл **log_2_4**, после чего были посчитаны средние значения, которые отображены на Рис.8.

Вывод

Эксперимент №1

- Все 4 графика показали линейный рост;

- При использовании одного ядра и последовательном запуске время выполнения (с ростом значения N) увеличивалось, примерно, на время выполнения одного запуска (2-3 секунды);
- При использовании одного ядра и параллельном запуске время выполнения (с ростом значения N) увеличивалось аналогично. Никакого выигрыша параллелизм не дал;
- При использовании двух ядер и последовательном запуске время выполнения (с ростом значения N) увеличивалось аналогично. Увеличение числа ядер не дало выигрыша при последовательном выполнение программы;
- При использовании двух ядер и параллельном запуске время выполнения (с ростом значения N) увеличивалось вдвое меньше, чем в предыдущих разах. Параллельные запуски с большим количеством ядер дают выигрыш во времени.

Эксперимент №2

- Все 4 графика показали линейный рост;
- При использовании одного ядра и последовательном запуске время выполнения (с ростом значения N) увеличивалось, примерно, на время выполнения одного запуска (2-3 секунды);
- При использовании одного ядра и параллельном запуске время выполнения (с ростом значения N) оказалось даже больше, чем при последовательном запуске. Параллельный запуск оказался хуже при 1 ядре;
- При использовании двух ядер и последовательном запуске время выполнения (с ростом значения N) уменьшилось. Увеличение ядер снизило время выполнения программы;
- При использовании двух ядер и параллельном запуске время выполнения (с ростом значения N) уменьшилось еще больше. Тем самым, получили значительный выигрыш во времени выполнения программы при использовании большего количества ядер и параллелизма.

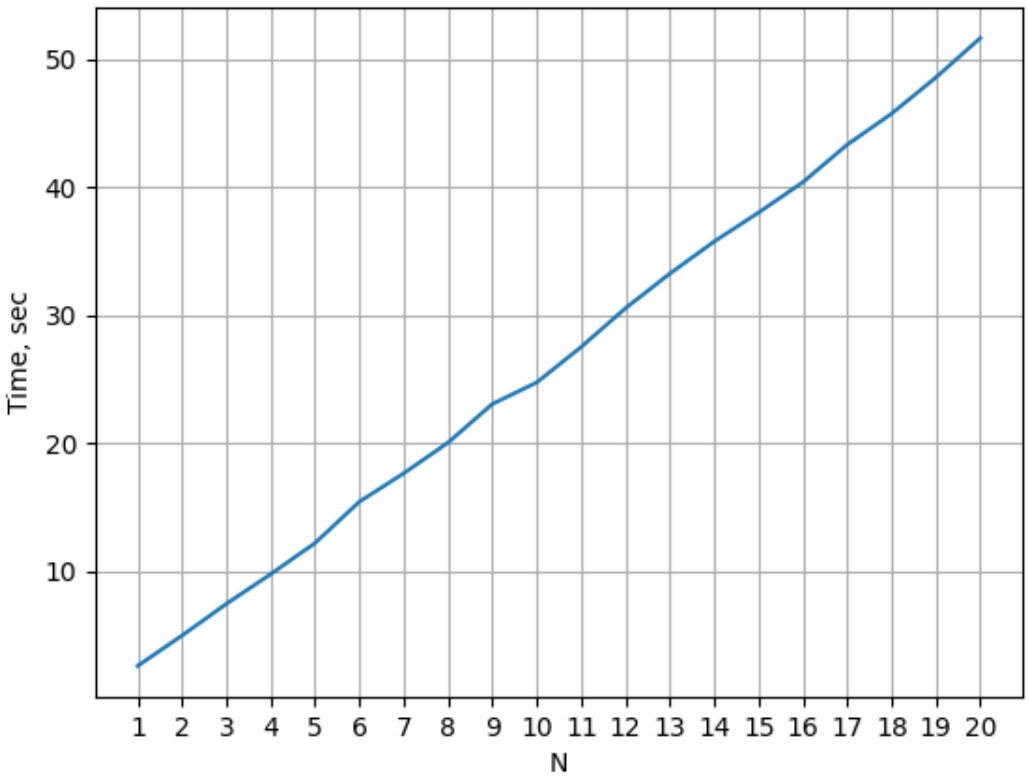

Рис. 1: Эксперимент №1, последовательный запуск, 1 ядро

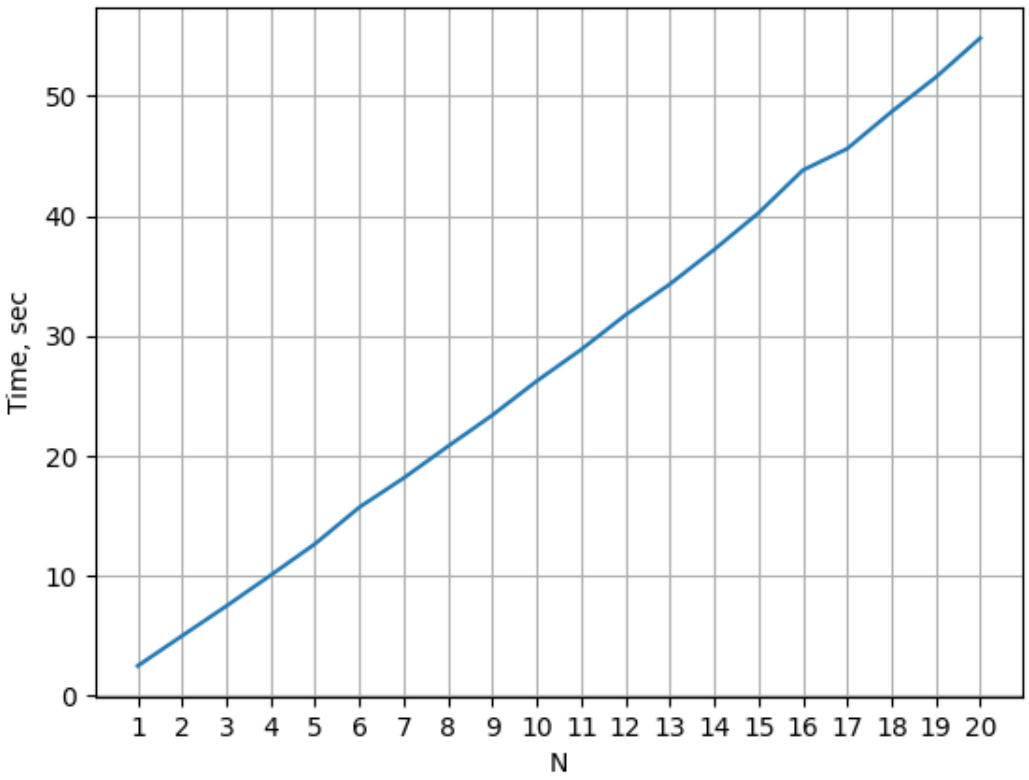

Рис. 2: Эксперимент №1, параллельный запуск, 1 ядро

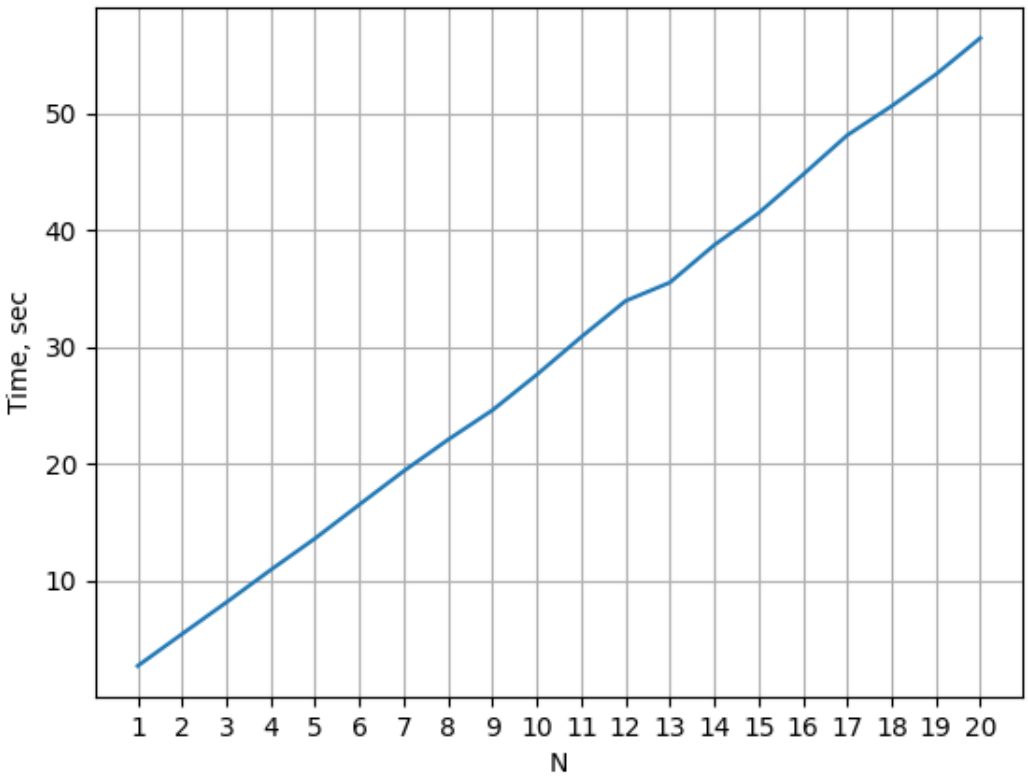

Рис. 3: Эксперимент №1, последовательный запуск, 2 ядра

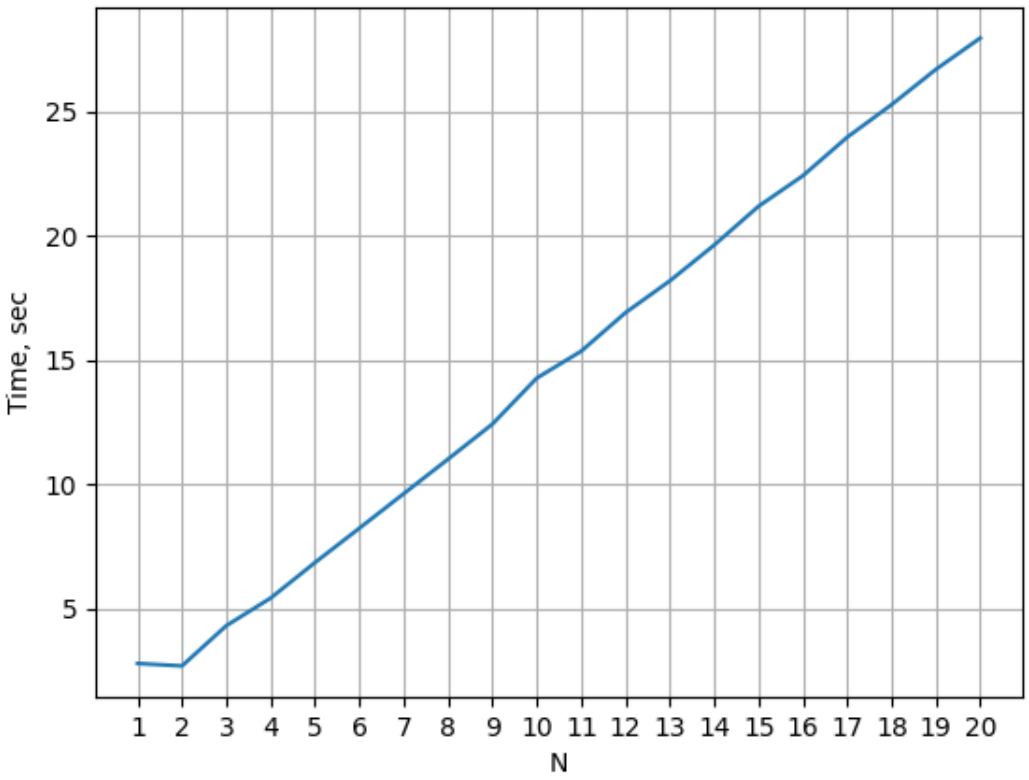

Рис. 4: Эксперимент №1, параллельный запуск, 2 ядра

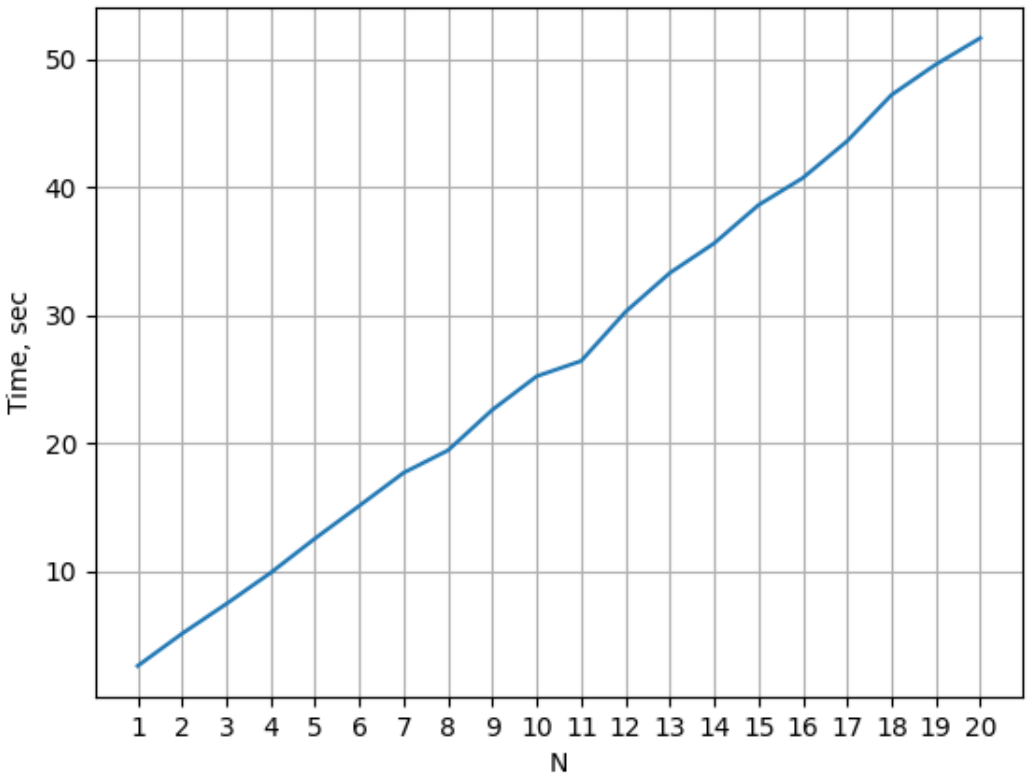

Рис. 5: Эксперимент №2, последовательный запуск, 1 ядро

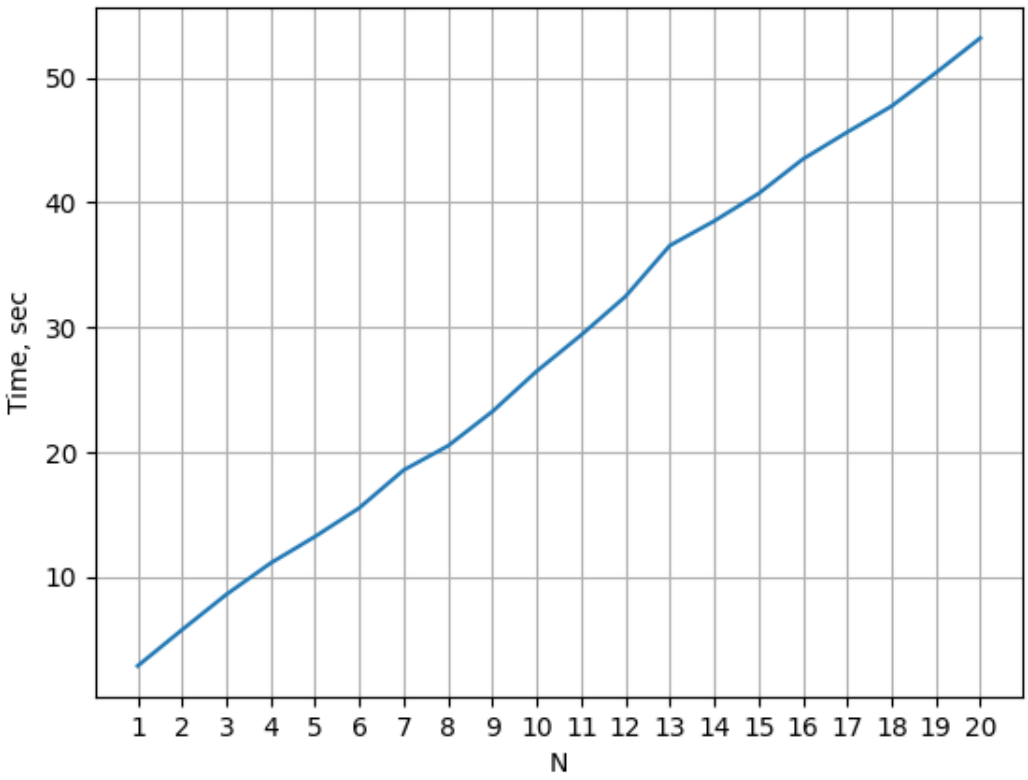

Рис. 6: Эксперимент №2, параллельный запуск, 1 ядро

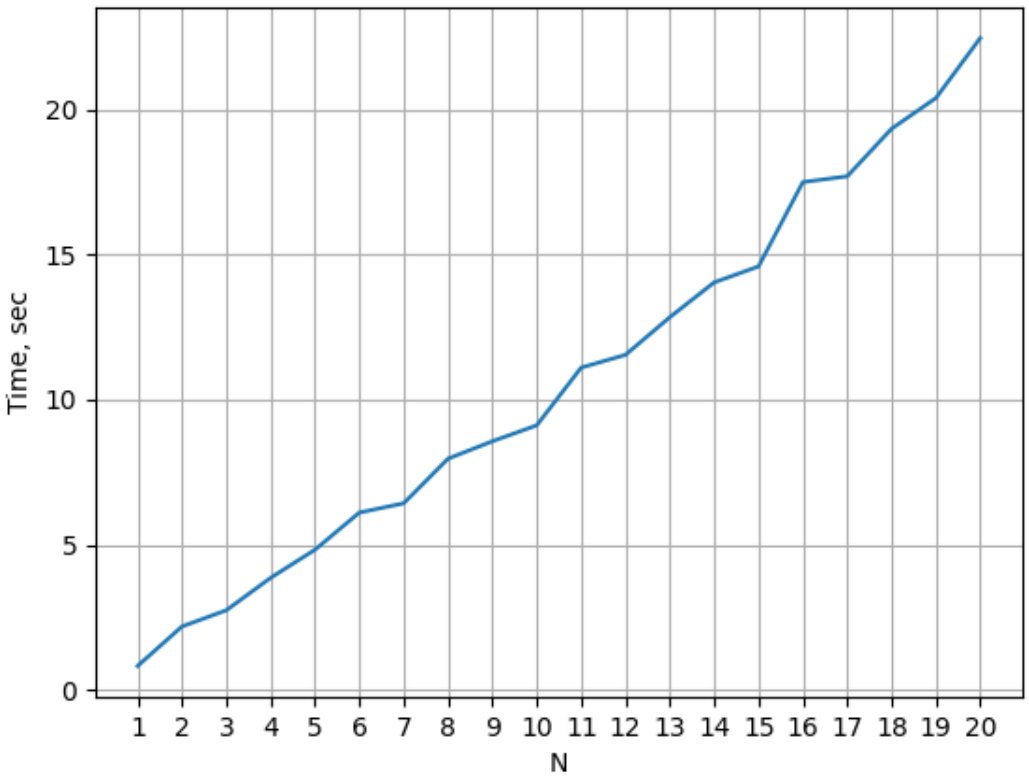

Рис. 7: Эксперимент №2, последовательный запуск, 2 ядра

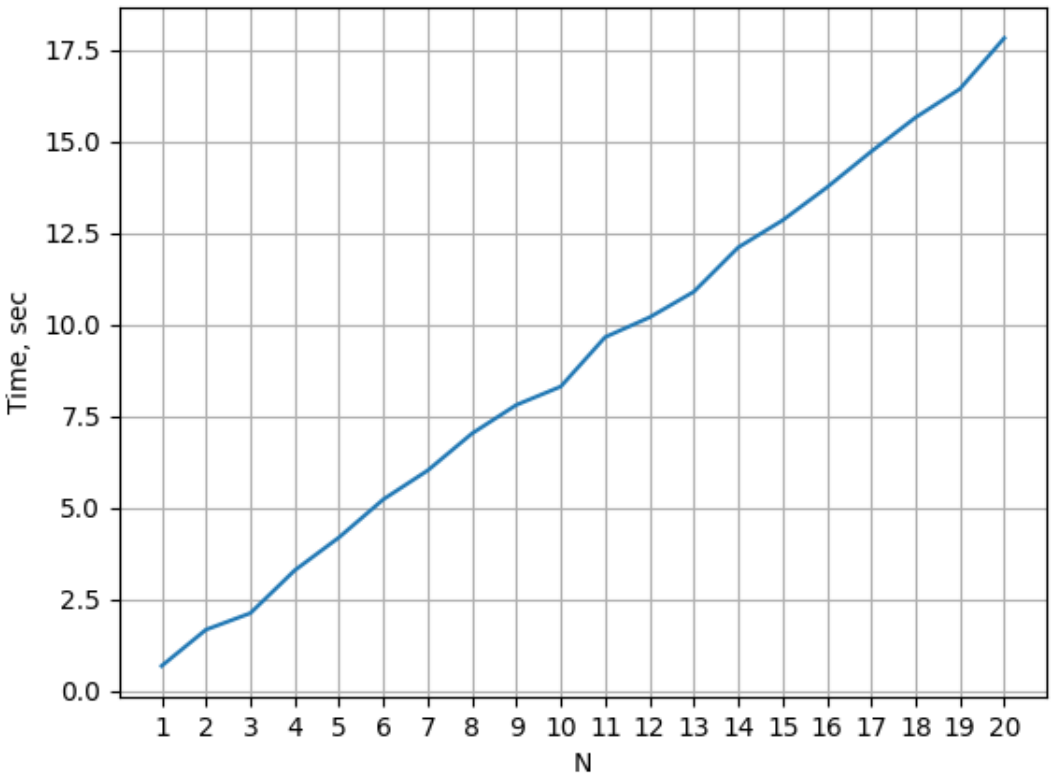

Рис. 8: Эксперимент №2, параллельный запуск, 2 ядра