



Факультет программной инженерии и компьютерной техники  
Параллельные вычисления

Лабораторная работа №2  
Исследование эффективности параллельных библиотек для C-  
программ

Преподаватель: Жданов Андрей Дмитриевич  
Выполнил: студент: Кульбако Артемий Юрьевич, Р4115

Санкт-Петербург  
2022

## **ОГЛАВЛЕНИЕ**

ОГЛАВЛЕНИЕ .....	2
ЗАДАНИЕ.....	3
ВЫПОЛНЕНИЕ.....	4
ИСХОДНЫЙ КОД.....	7

## ЗАДАНИЕ

1. В исходном коде программы, полученной в результате выполнения лабораторной работы №1, нужно на этапах Map и Merge все циклы с вызовами математических функций заменить их векторными аналогами из библиотеки «AMD Framewave» (<http://framewave.sourceforge.net>). При выборе конкретной Framewave-функции необходимо убедиться, что она помечена как MT (Multi-Threaded), т.е. распараллеленная. Полный перечень доступных функций находится по ссылке: [http://framewave.sourceforge.net/Manual/fw\\_section\\_060.html#fw\\_section\\_060](http://framewave.sourceforge.net/Manual/fw_section_060.html#fw_section_060). Например, Framewave-функция min в списке поддерживаемых технологий имеет только SSE2, но не MT.

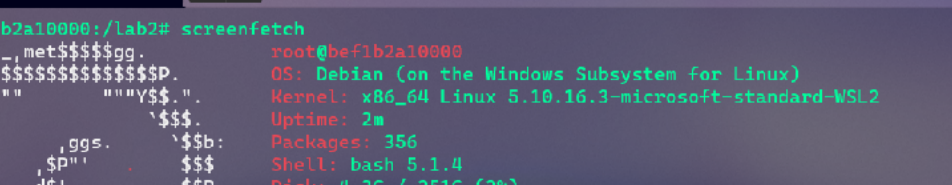
Примечание: выбор библиотеки Framewave не является обязательным, можно использовать любую другую параллельную библиотеку, если в ней нужные функции распараллелены, так, например, можно использовать ATLAS (для этой библиотеки необходимо выключить троттлинг и энергосбережение, а также разобраться с механизмом изменения числа потоков) или Intel Integrated Performance Primitives.

2. Добавить в начало программы вызов Framewave-функции SetNumThreads(M) для установки количества создаваемых параллельной библиотекой потоков, задействуемых при выполнении распараллеленных Framewave-функций. Нужно число M следует устанавливать из параметра командной строки (argv) для удобства автоматизации экспериментов.

Примечание: При использовании Intel IPP функцию SetNumThreads(M) не нужно использовать. Необходимо компилировать программу под разное количество потоков.

3. Скомпилировать программу, не применяя опции автоматического распараллеливания, использованные в лабораторной работе №1. Провести эксперименты с полученной программой для тех же значений  $N_1$  и  $N_2$ , которые использовались в лабораторной работе №1, при  $M = 1, 2, \dots, K$ , где  $K$  – количество процессоров (ядер) на экспериментальном стенде.
4. Сравнить полученные результаты с результатами лабораторной работы №1: на графиках показать, как изменилось время выполнения программы, параллельное ускорение и параллельная эффективность.
5. Написать отчёт о проделанной работе.
6. Подготовиться к устным вопросам на защите.

## ВЫПОЛНЕНИЕ



```
Windows PowerShell
root@bef1b2a10000:/lab2# screenfetch

_,met$$$$$gg.
g$$$$$$$$$$$$P.
g$P" " "Y$.
'$P' '$$.
'$P' ggs. '$$b:
'd$$' '$P' '$$
'$P' d'$' '$P
$$: $$ - d$$'
$$. $b._ dP'
Y$. \.'Y$$$$P'
'$$b "-__
'Y$$
'Y$.
'$$b.
'Y$$b.
'Y$b._
      .

root@bef1b2a10000

OS: Debian (on the Windows Subsystem for Linux)
Kernel: x86_64 Linux 5.10.16.3-microsoft-standard-WSL2
Uptime: 2m
Packages: 356
Shell: bash 5.1.4
Disk: 4.2G / 251G (2%)
CPU: AMD Ryzen 7 2700X Eight-Core @ 16x 3.693GHz
RAM: 859MiB / 15960MiB

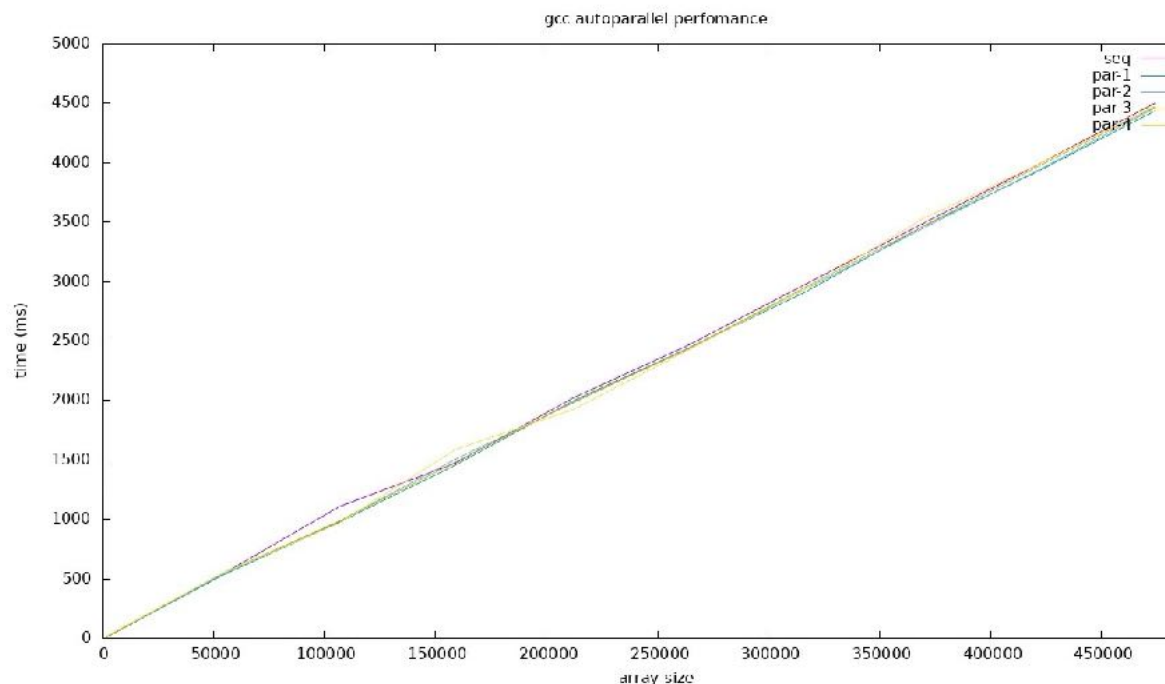
root@bef1b2a10000:/lab2#
```

## Характеристики виртуальной машины (Docker)

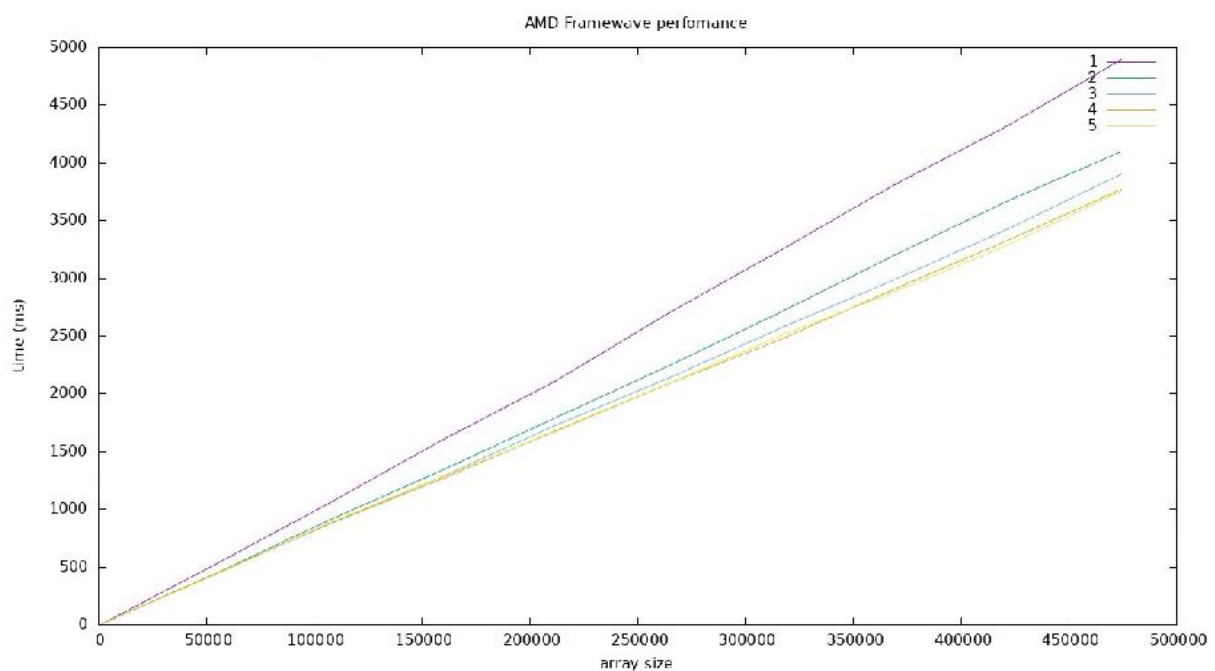
В качестве параллельной библиотеки была выбрана AMD Framewave, а в качестве компилятора gcc.

В результате выполнения прошлой лабораторной работы было видно, что автораспараллеливание компилятора не дало

никакого результата. Библиотека Framewave сработала лучше, видно, что ускорение есть: видно, что работа с 450000 элементов в первом случае заняла чуть больше 4 сек, у Framewave на это ушло менее 3.5 сек, то есть ускорение где-то 12.5%.



Автоматическое распараллеливание gcc

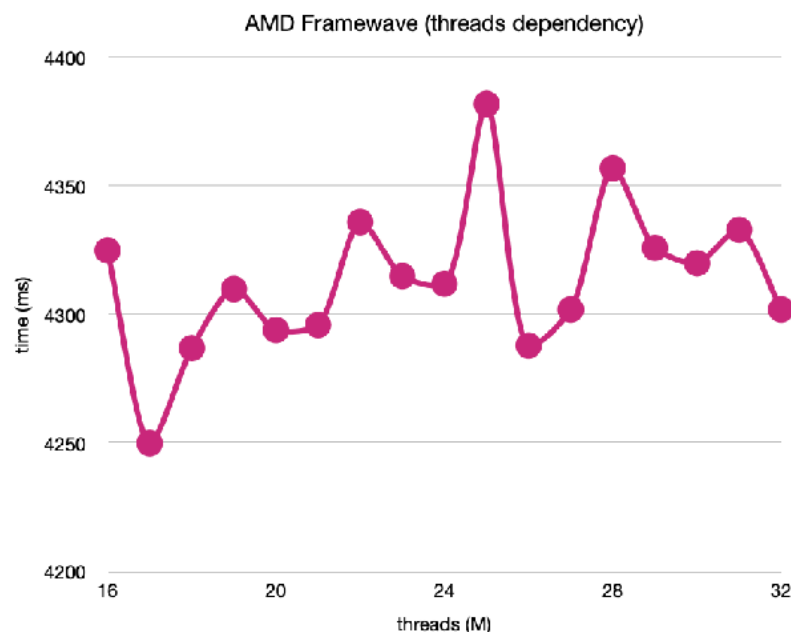


Распараллеливание средствами AMD Framewave

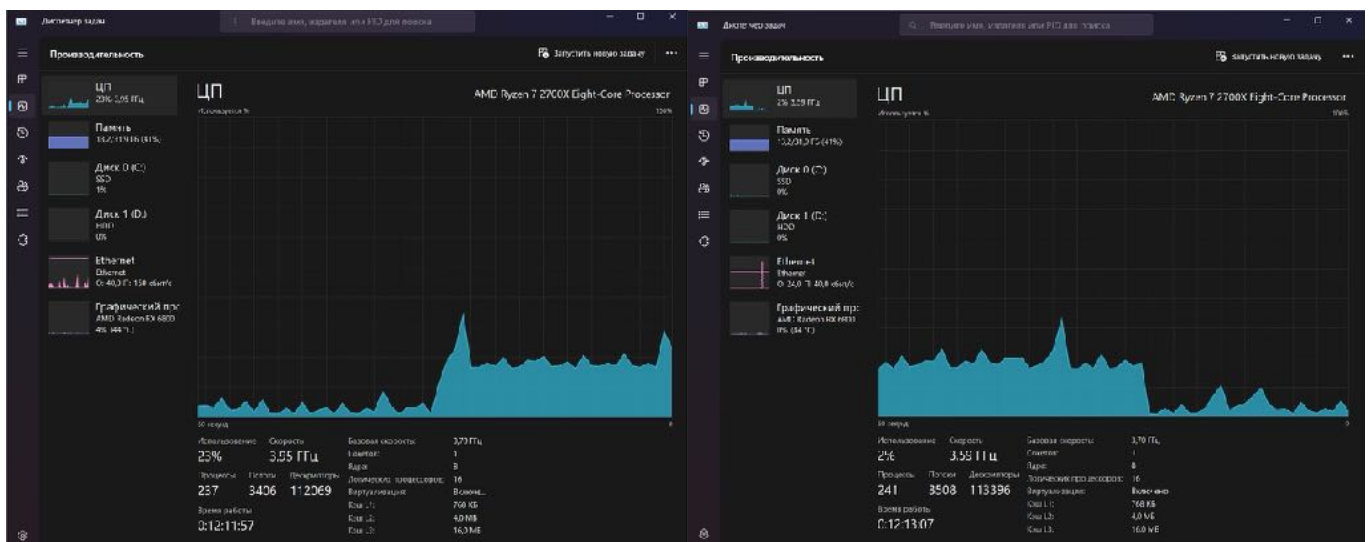
При тестировании программы с  $N_2 = 526999$  и разным количеством потоков происходят скачки времени выполнения: можно сделать вывод о том, что накладные ресурсы на создание потоков в данном эксперименте неинтрузивно.

amd\_fw4

16	4325
17	4250
18	4287
19	4310
20	4294
21	4296
22	4336
23	4315
24	4312
25	4382
26	4288
27	4302
28	4357
29	4326
30	4320
31	4333
32	4302



Графики загрузки процессора в начале и в конце тестирования соответственно:



Начало тестирования

Конец тестирования

## ИСХОДНЫЙ КОД

Таблицы в формате csv и исходной код программы доступен на:  
<https://github.com/testpassword/Parallel-computing/tree/master/lab2-18.03.23>

