Федеральное государственное автономное образовательное учреждение высшего образования

«**Университет ИТМО»**

Факультет ПИиКТ

Дисциплина: Параллельные вычисления

Лабораторная работа 1

Вариант

A = 280

Гиперболический котангенс корня числа

Логарифм по 10 основанию в степени e

Выбор большего (т.е. M2[i] = max(M1[i],M2[i])))

Глупая сортировка (Stupid sort).

Выполнил: Гурин Евгений Иванович

Преподаватель: Жданов Андрей Дмитриевич

Группа: P4116

Санкт-Петербург 2023г.

Задача

1. На компьютере с многоядерным процессором установить ОС Linux и компилятор GCC версии не ниже 4.7.2. При невозможности установить Linux или отсутствии компьютера с многоядерным процессором можно выполнять лабораторную работу на виртуальной машине. Минимальное количество ядер при использовании виртуальной машины - 2.
2. На языке Си написать консольную программу lab1.c, решающую задачу, указанную в п.IV (см. ниже). В программе нельзя использовать библиотечные функции сортировки, выполнения матричных операций и расчёта статистических величин. В программе нельзя использовать библиотечные функции, отсутствующие в стандартных заголовочных файлах stdio.h, stdlib.h, math.h, sys/time.h. Задача должна решаться 100 раз с разными начальными значениями генератора случайных чисел (ГСЧ).
3. Скомпилировать написанную программу без использования автоматического распараллеливания с помощью следующей команды: /home/user/gcc -O3 -Wall -Werror -o lab1-seq lab1.c
4. Скомпилировать написанную программу, используя встроенное в gcc средство автоматического распараллеливания Graphite с помощью следующей команды “/home/user/gcc -O3 -Wall -Werror -floop-parallelize-all -ftree-parallelize-loops=K lab1.c -o lab1-par-K” (переменной K поочерёдно присвоить хотя бы 4 значения: 1, меньше количества ядер, равное количеству физических ядер и больше количества физических ядер).
5. В результате получится одна нераспараллеленная программа и четыре или более распараллеленных.
6. Закрыть все работающие в операционной системе прикладные программы (включая Winamp, uTorrent, браузеры и Skype), чтобы они

не влияли на результаты последующих экспериментов. При использовании ноутбука необходимо иметь постоянное подключение к сети питания, на время проведения эксперимента.

7. Запускать файл lab1-seq из командной строки, увеличивая значения N до значения N1, при котором время выполнения превысит 0.01 с. Подобным образом найти значение N=N2, при котором время выполнения превысит 5 с.

8. Используя найденные значения N1 и N2, выполнить следующие эксперименты (для автоматизации проведения экспериментов рекомендуется написать скрипт)

10. Написать отчёт о проделанной работе.

11. Подготовиться к устным вопросам на защите.

12. Найти вычислительную сложность алгоритма до и после распараллеливания, сравнить полученные результаты.

13. Необязательное задание №1 (для получения оценки «четыре» и «пять»). Провести аналогичные описанным эксперименты, используя вместо gcc компилятор Solaris Studio (или любой другой на своё усмотрение). При компиляции следует использовать следующие опции для автоматического распараллеливания: «`clang -O3 -Wall -Werror -mllvm -force-vector-width={i} main.c -o builded-clang/lab1-par-{i} -lm`».

14. Необязательное задание №2 (для получения оценки «пять»). Это задание выполняется только после выполнения предыдущего пункта. Провести аналогичные описанным эксперименты, используя вместо gcc компилятор Intel ICC (или любой другой на своё усмотрение). В ICC следует при компиляции использовать следующие опции для автоматического распараллеливания: «`tcc -O3 -Wall -Werror -floop-parallelize-all -ftree-parallelize-loops={i} main.c -o builded-tcc/lab1-par-{i} -lm`».

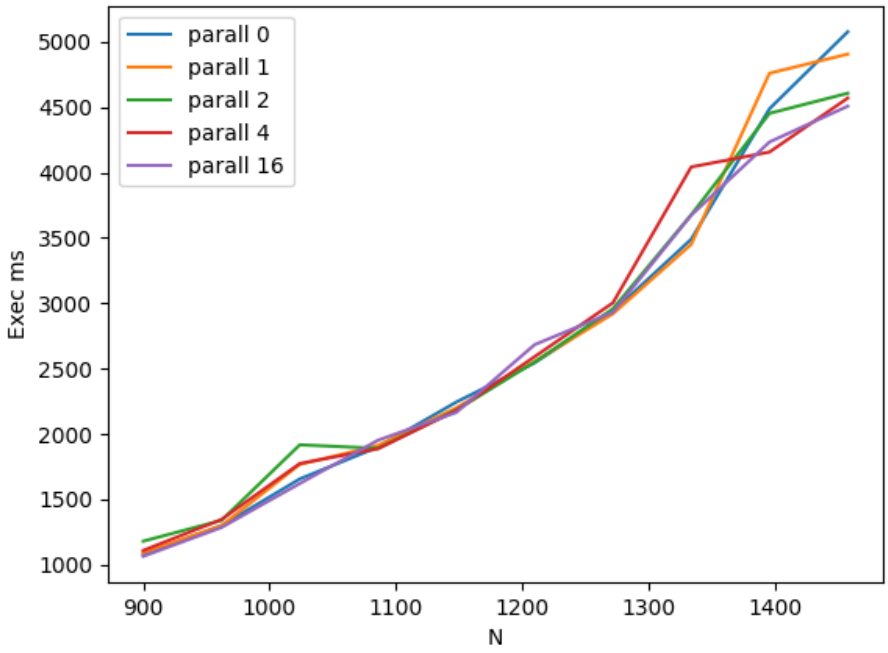
Конфигурация

```
Host Name:                 EGURIN-LAPTOP
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22000 N/A Build 22000
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          user
Registered Organization:   N/A
Product ID:                00331-10000-00001-AA468
Original Install Date:     18.09.2022, 18:05:53
System Boot Time:          16.02.2023, 20:08:15
System Manufacturer:       Timi
System Model:              TM1701
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 142 Stepping 10
GenuineIntel ~1801 Mhz
BIOS Version:              INSYDE Corp. XMAKB5R0P0603, 02.02.2018
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume2
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+03:00) Moscow, St. Petersburg
Total Physical Memory:     8 076 MB
Available Physical Memory: 752 MB
Virtual Memory: Max Size:  23 948 MB
Virtual Memory: Available: 11 975 MB
Virtual Memory: In Use:    11 973 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\EGURIN-LAPTOP
Hotfix(s):                 4 Hotfix(s) Installed.
                           [01]: KB5020875
                           [02]: KB5012170
                           [03]: KB5019961
                           [04]: KB5017850
Network Card(s):           2 NIC(s) Installed.
                           [01]: Intel(R) Dual Band Wireless-AC 8265
                                 Connection Name: Wi-Fi
                                 DHCP Enabled:    Yes
                                 DHCP Server:     192.168.13.254
                                 IP address(es)
                                 [01]: 192.168.13.179
                                 [02]: fe80::93ec:19f3:20d7:dd94
                           [02]: Bluetooth Device (Personal Area Network)
                                 Connection Name: Bluetooth Network Connection
                                 Status:          Media disconnected
Hyper-V Requirements:      A hypervisor has been detected. Features required for
Hyper-V will not be displayed.
```

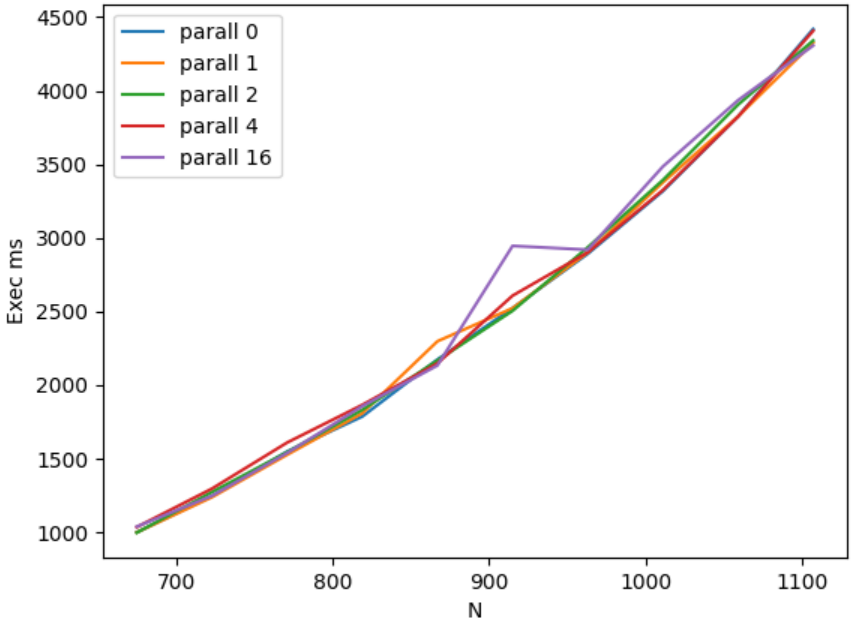
Результаты работы

Были выбраны 3 компилятора: GCC, Tiny C Compiler, clang

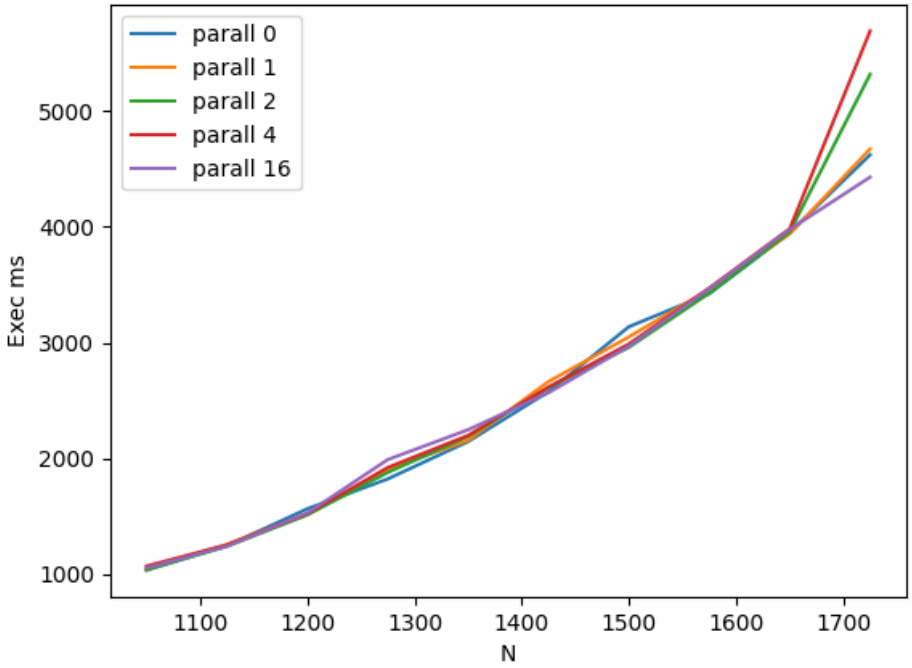
GCC



CLANG

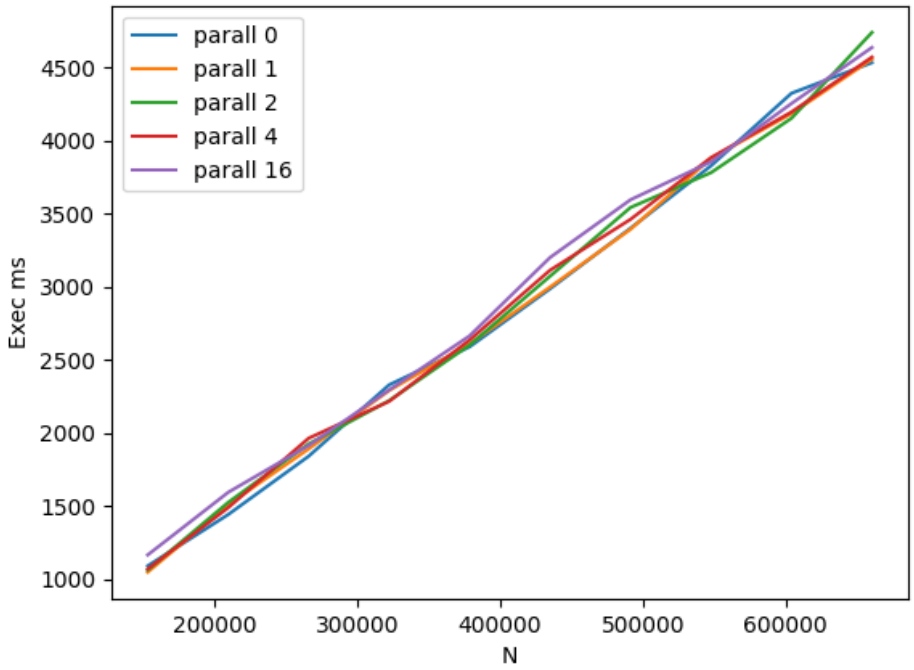


TCC

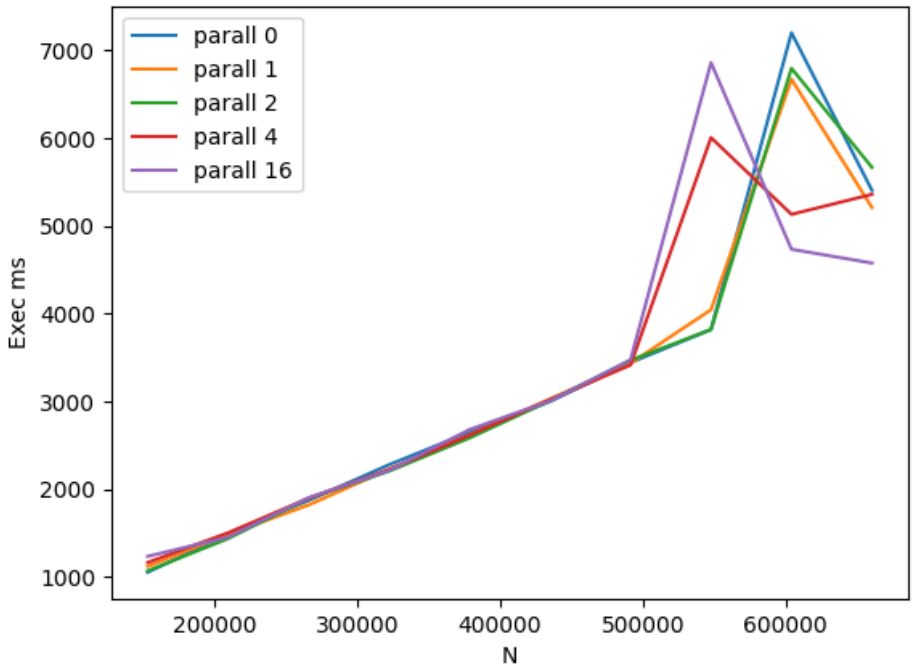

parall 0
parall 1
parall 2
parall 4
parall 16
Exec ms
5000
4000
3000
2000
1000
1100
1200
1300
1400
1500
1600
1700
N

Без сортировки

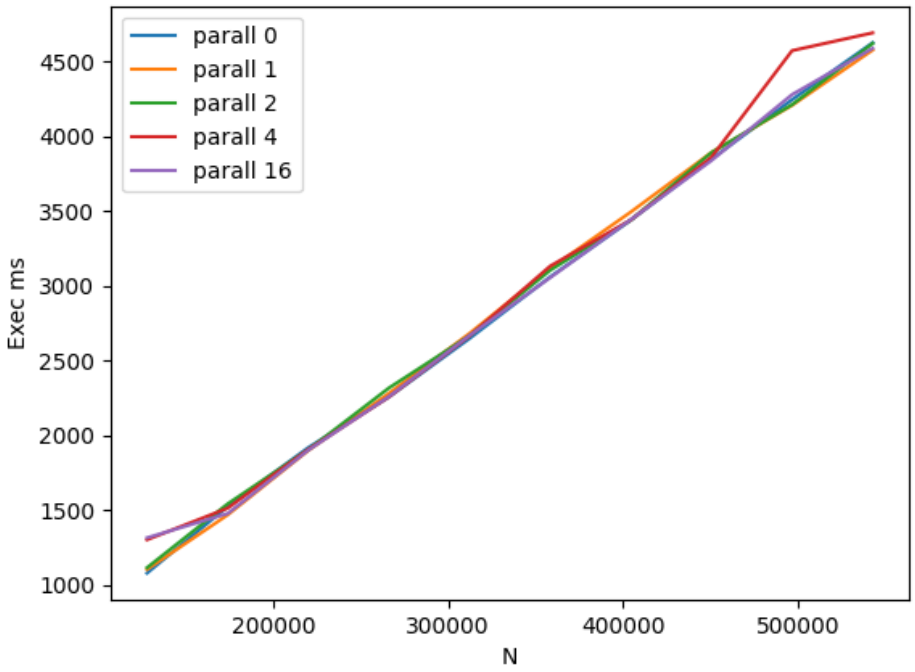
GCC



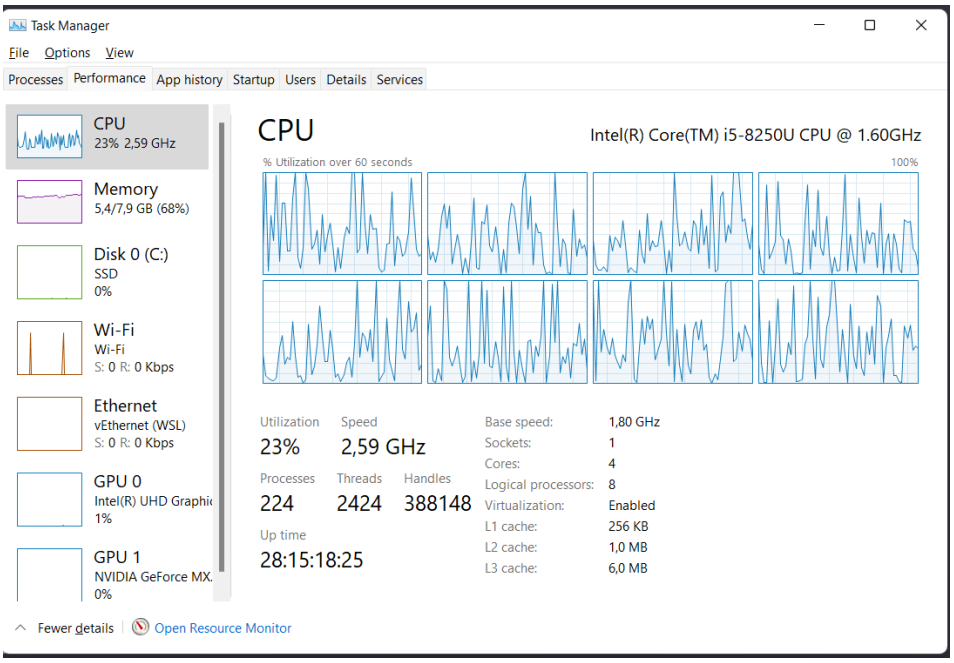
CLANG



TCC



Загрузка процессора



Листинг main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>


void swap(double *a, double *b) {
    double t;
    t = *a, *a = *b, *b = t;
}


void sort_stupid(double *array, int n) {
    int i = 0;
    while (i < n - 1) {
        if (array[i + 1] < array[i]) swap(array + i, array + i + 1), i = 0;
        else i++;
    }
}

void print_arr(double *array, int n) {
    for (int i = 0; i < n; ++i)
    {
        printf("%f ", array[i]);
    }
    printf("\n");
}

int main(int argc, char *argv[]) {
    unsigned int N, N_2;
    struct timeval T1, T2;
    long delta_ms;
    // N = 4000;
    gettimeofday(&T1, NULL); /* запомнить текущее время T1 */

    for (unsigned int i = 0; i < 100; i++) /* 100 экспериментов */
    {
        N = atoi(argv[1]); /* N равен первому параметру командной строки */
        N_2 = N / 2;
        double * restrict m1 = malloc(N * sizeof(double));
        double * restrict m2 = malloc(N_2 * sizeof(double));
        double * restrict m2_cpy = malloc(N_2 * sizeof(double));
        int A = 280;

        unsigned int seedp = i;
```

```
        // generate 1
        for (int j = 0; j < N; ++j) {

            m1[j] = (rand_r(&seedp) % (A * 100)) / 100.0 + 1;
        }

        // generate 2
        for (int j = 0; j < N_2; ++j) {
            m2[j] = A + rand_r(&seedp) % (A * 9);
        }
        for (int j = 0; j < N_2; ++j) {
            m2_cpy[j] = m2[j];
        }

        // map
        for (int j = 0; j < N; ++j) {
            m1[j] = 1 / tanh(sqrt(m1[j]));
        }
        for (int j = 1; j < N_2; ++j) {
            m2[j] = m2[j] + m2_cpy[j - 1];
        }
        for (int j = 1; j < N_2; ++j) {
            m2[j] = pow(log10(m2[j]), M_E);
        }

        for (int j = 0; j < N_2; ++j) {
            m2[j] = m2[j] > m1[j] ? m2[j] : m1[j] ;
        }

        // sort_stupid(m2, N_2);

        // reduce
        double X = 0;
        int k = 0;
        while (m2[k] == 0 && k < N_2 - 1) k++;
        double m2_min = m2[k];

        for (int j = 0; j < N_2; ++j) {
            if((int)(m2[j] / m2_min) % 2 == 0) X += sin(m2[j]);
        }
        printf("%f ", X);
    }
    gettimeofday(&T2, NULL);
    /* запомнить текущее время T2 */
    delta_ms = 1000 * (T2.tv_sec - T1.tv_sec) + (T2.tv_usec - T1.tv_usec) /
1000;
    printf("\n%ld\n", delta_ms); /* T2 - T1 */
```

```
    return 0;
}
```

Вывод

Все компиляторы не показали прироста от параллелизма. Это можно связать с тем, что асимптотика глупой сортировки N^3 и время его работы много превышает время, которое может быть выиграно при распараллеливании возможных участков. Для различных компиляторов есть различия во времени исполнения программы, самый быстрый – clang

В случае без сортировки результаты более интересные. Можно заметить, что график для GCC идёт практически идентично и похоже оптимизация не принесла результатов, хотя потенциальные участки для распараллеливания имеются. Для clang на более крупных объёмах данных было различие в результатах и виден прирост для многопоточных сборок на последних экспериментах. Для компилятора TCC прироста не наблюдается. Большинство циклов содержат зависимости в виде использования функций, и компилятор вероятно распараллелить их не может. Верификацию результатов при этом прошли все эксперименты.