Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет Программной инженерии и компьютерной техники

Направление подготовки (специальность) Системное и прикладное ПО

ОТЧЕТ

Лабораторная работа №1
по предмету «Параллельные вычисления»

Тема проекта: «Автоматическое распараллеливание программ».

Обучающийся Ткаченко В.В. (**Фамилия И.О.**) Р4114 (**номер группы**)

Преподаватель Жданов А. Д. (**Фамилия И.О.**)

Санкт-Петербург
2023 г.

Содержание

Описание решаемой задачи

На языке Си написать консольную программу lab1.c, решающую задачу, указанную ниже, согласно вариантам: A=576, Map=6|5, Merge = 1, Sort=6.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).

1. Этап Generate. Сформировать массив M1 размерностью N, заполнив его с помощью функции rand_r (нельзя использовать rand) случайными вещественными числами, имеющими равномерный закон распределения в диапазоне от 1 до A (включительно). Аналогично сформировать массив M2 размерностью N/2 со случайными вещественными числами в диапазоне от A до 10*A.
2. Этап Map. В массиве M1 к каждому элементу применить операцию: Кубический корень после деления на число e. Затем в массиве M2 каждый элемент поочерёдно сложить с предыдущим (для этого вам понадобится копия массива M2, из которого нужно будет брать операнды), а к результату сложения применить операцию: Натуральный логарифм модуля тангенса.
3. Этап Merge. В массивах M1 и M2 ко всем элементами с одинаковыми индексами попарно применить операцию: Возведение в степень (т.е. M2[i] = M1[i]^M2[i])
4. Этап Sort. Полученный массив необходимо отсортировать методом: сортировка вставками.
5. Этап Reduce. Рассчитать сумму синусов тех элементов массива M2, которые при делении на минимальный ненулевой элемент массива M2 дают чётное число (при определении чётности учитывать только целую

часть числа). Результатом работы программы по окончании пятого этапа должно стать одно число X, которое следует использовать для верификации программы после внесения в неё изменений (например, до и после распараллеливания итоговое число X не должно измениться в пределах погрешности). Данное число необходимо выводить на каждой итерации на этапе верификации. Значение числа X следует привести в отчёте для различных значений N.

Краткая характеристика системы

Операционная система: ubuntu 22.04.2 LTS on Win 10 x86_64

Оперативная память: 16гб

Процессор: intel i7-4710MQ

Кол-во физических ядер: 4

Кол-во логических ядер: 8

Семейство процессоров: core i7

Версия GCC: 11.3.0

программы lab1.c

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

void insertion_sort(double arr[], int n)
{
    int i, j;
    double key;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
```

```
}

int main(int argc, char *argv[])
{
    int i, j, N;
    double e = exp(1.0);
    struct timeval T1, T2;
    long delta_ms;
    double min = 1;
    double max = 576;

    N = atoi(argv[1]);

    double *restrict M1 = (double *)malloc(N * sizeof(double));
    double *restrict M2 = (double *)malloc(N / 2 * sizeof(double));
    double *restrict M2_copy = (double *)malloc(N / 2 * sizeof(double));

    unsigned int seed;
    unsigned int *restrict seed1 = &seed;
    unsigned int *restrict seed2 = &seed;

    gettimeofday(&T1, NULL);

    for (i = 0; i < 100; i++)
    {
        seed = i;

        for (j = 0; j < N; j++)
        {
            // делим каждый член на e и считаем кубический корень
            M1[j] = cbrt((((double)rand_r(seed1) / (RAND_MAX)) * (max - min) + min) /
e);
        }
        for (j = 0; j < N / 2; j++)
        {

            M2[j] = ((double)rand_r(seed2) / (RAND_MAX)) * (max * 10 - max) + max;
            M2_copy[j] = M2[j];
        }
        for (j = 1; j < N / 2; j++)
        {
            M2[j] = log(fabs(tan(M2[j] + M2_copy[j - 1])));
        }
        M2[0] = log(fabs(tan(M2[0])));

        for (j = 0; j < N / 2; j++)
        {
            M2[j] = pow(M1[j], M2[j]);
        }
        insertion_sort(M2, N / 2);
        double min_nonzero = INFINITY;
        double sum_sin = 0.0;
        for (j = 0; j < N / 2; j++)
        {
            if (M2[j] > 0 && M2[j] < min_nonzero)
            {
                min_nonzero = M2[j];
            }
            if ((int)(M2[j] / min_nonzero) % 2 == 0)
            {
                sum_sin += sin(M2[j]);
```

```
            }
        }
        printf("%lf\n", sum_sin);
    }
    // printf("%lf\n", sum_sin);
    gettimeofday(&T2, NULL);
    delta_ms = 1000 * (T2.tv_sec - T1.tv_sec) + (T2.tv_usec - T1.tv_usec) / 1000;
    printf("\nN=%d. Milliseconds passed: %ld\n", N, delta_ms);

    free(M1);
    free(M2);
    free(M2_copy);

    return 0;
}
```

Результаты экспериментов

N	ms		seq	par-1	par-2	par-4	par-5
1000	500		10	12	11	10	11
4700	1000		109	109	109	109	108
8400	1500		297	318	311	298	295
12100	2000		575	594	624	578	581
15800	2500		946	1080	980	948	949
19500	3000		1398	1536	1431	1416	1478
23200	3500		1952	2079	2057	1965	2103
26900	4000		2606	2663	2600	2627	2763
30600	4500		3346	3338	3341	3341	3375
34300	5000		4171	4148	4176	4237	4165
38000	5500		5122	5087	5089	5067	5077

Значения экспериментальных данных при компилляции программы с помощью gcc.

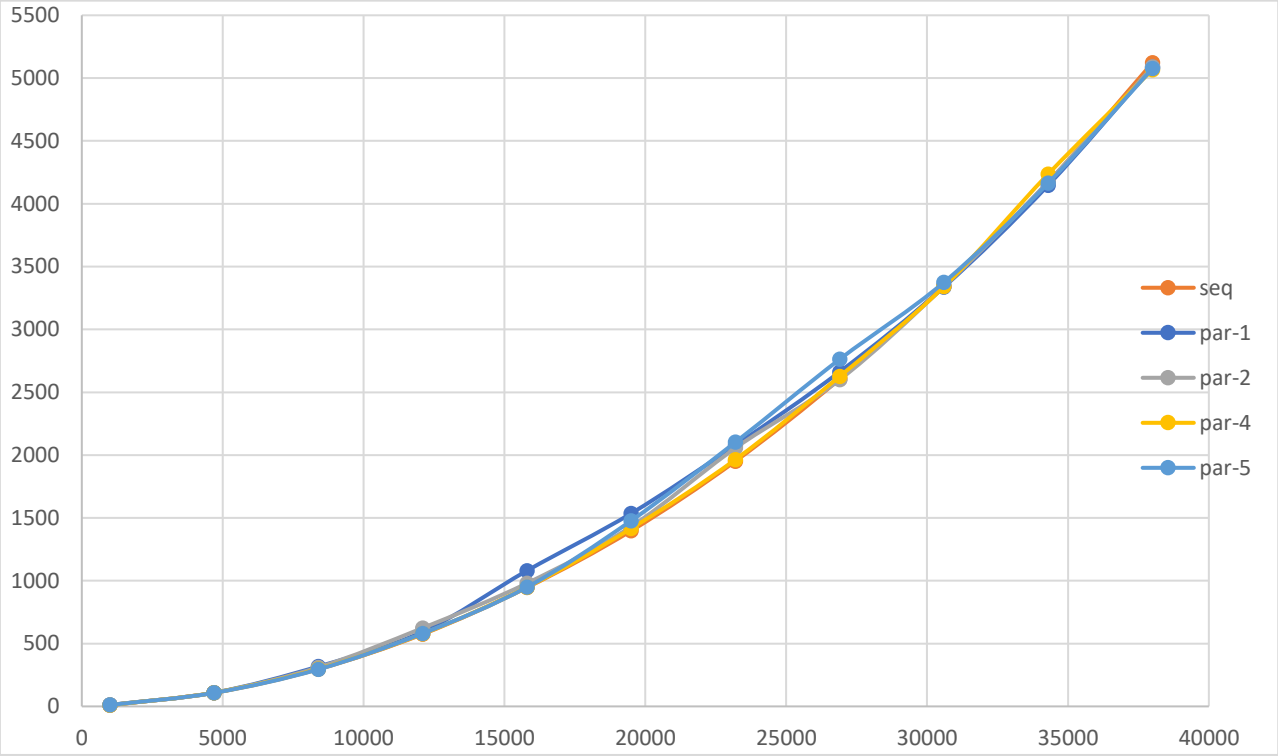
Ниже приведен получившийся график. Распараллеливание при использовании компилятора gcc не даёт каких-либо ощутимых изменений во времени выполнения программы, ускорения не наблюдается.

```
1558.164817
1663.769920
1583.714855

N=20000. Milliseconds passed: 1514
```

Величина X для нераспараллеленой программы

```
1558.164817
1663.769920
1583.714855

N=20000. Milliseconds passed: 1494
```

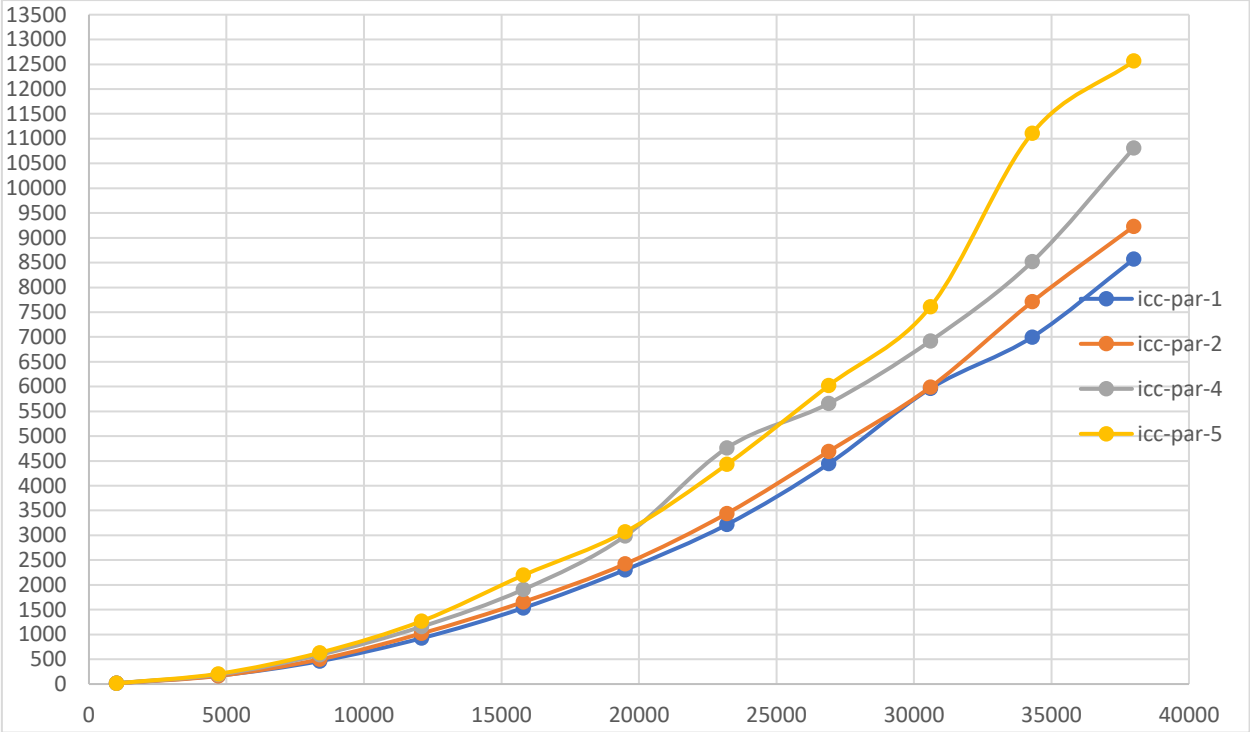
Величина X при распараллеливании на 5 потоков



Также были проведены эксперименты с помощью компилятора icc. В ходе обработки данных, было зафиксировано, что при распараллеливании программы, ускорения не наблюдается, а наоборот увеличивается время выполнения. Табличные значения и график представлены ниже.

N	ms		icc-par-1	icc-par-2	icc-par-4	icc-par-5
1000	500		17	16	18	15
4700	1000		169	164	190	207
8400	1500		463	499	591	633
12100	2000		925	1022	1158	1270
15800	2500		1535	1656	1908	2196
19500	3000		2304	2422	2985	3069
23200	3500		3217	3441	4762	4431
26900	4000		4443	4693	5659	6021
30600	4500		5962	5990	6921	7609

34300	5000		6998	7710	8518	11110
38000	5500		8574	9232	10812	12565



Вывод

В рамках лабораторной работы было рассмотрено автоматическое распараллеливание программы с использованием компилляторов gcc и icc. Выполнение программы как без распараллеливания, так и с ним отличается по скорости выполнения в пределах погрешности. При использовании компилятора icc на моей системе наблюжается тенденция к увеличению времени времени выполнения программы.