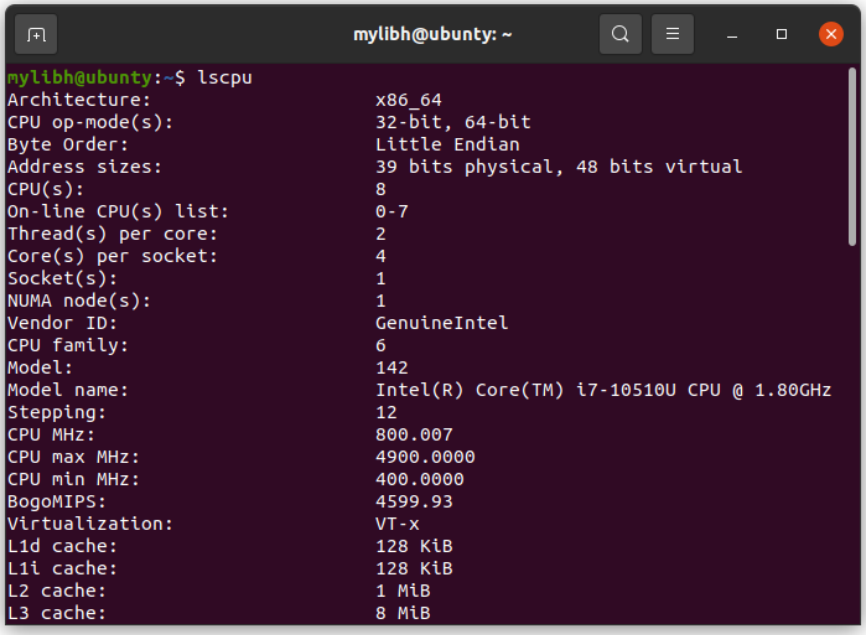
Параллельное программирование
Лабораторная работа №1

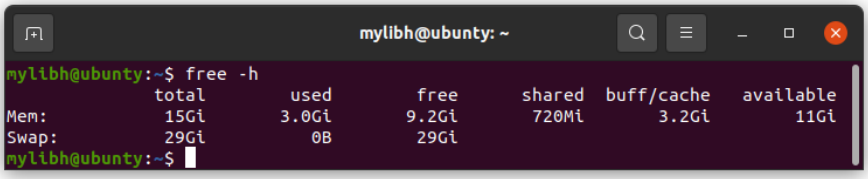
ИИКС ИБ
Б19-505
Голигузов Алексей

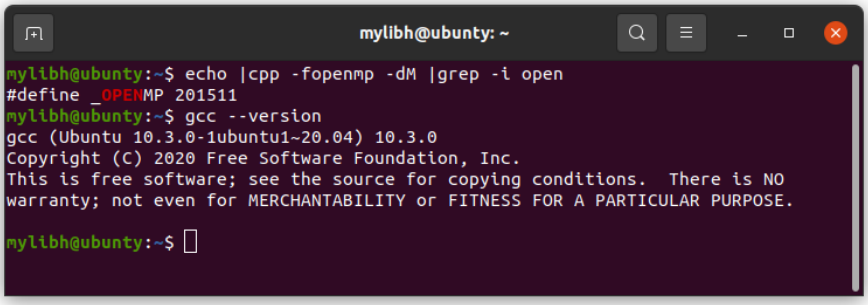
Сентябрь 2021

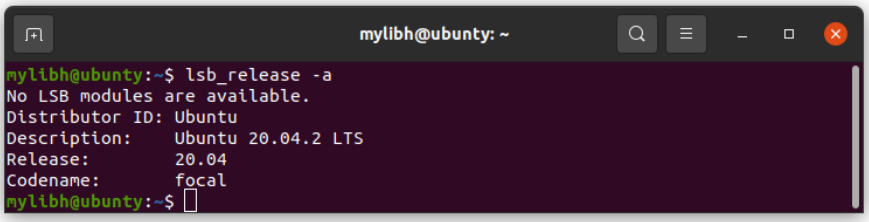
Содержание

1 Рабочая среда








mylibh@ubunty: ~
mylibh@ubunty:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 20.04.2 LTS
Release: 20.04
Codename: focal
mylibh@ubunty:~$

2 Анализ

Алгоритм работает за $O\left(\frac{n}{N}\right)$, где n - кол-во данных, N - кол-во потоков.

```
#pragma omp parallel for num_threads(threads_num) reduction(max: max)
```

Распараллеливаем цикл на $threads\_num$ потоков, в конце собираем в max локальные max со всех потоков.

3 Графики

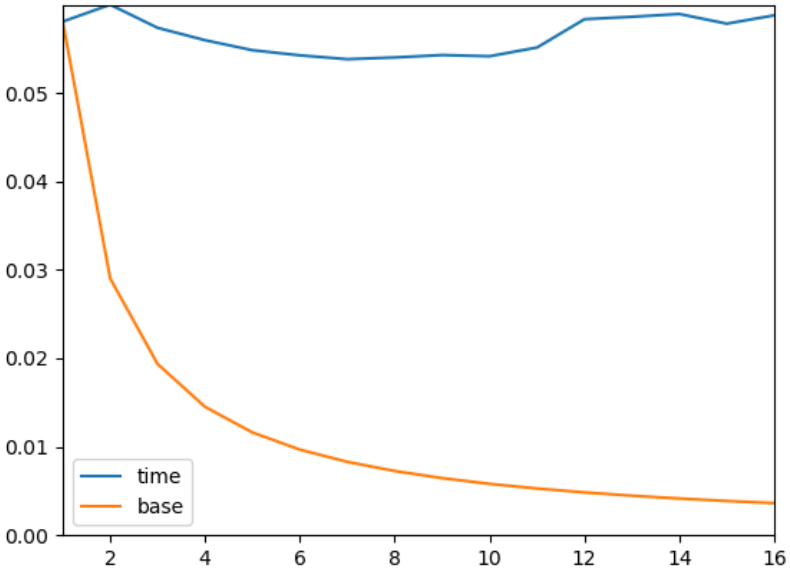


Рис. 1: Время

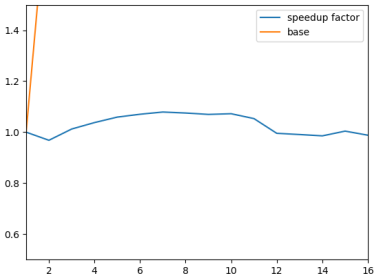


Рис. 2: Ускорение

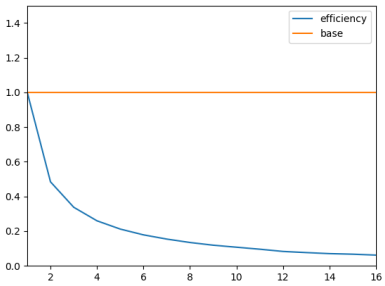


Рис. 3: Эффективность

4 Заключение

Время

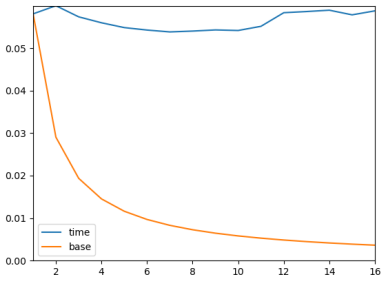


Рис. 4: Разные

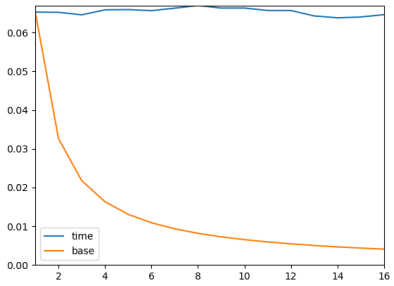


Рис. 5: Одинаковые

Ускорение

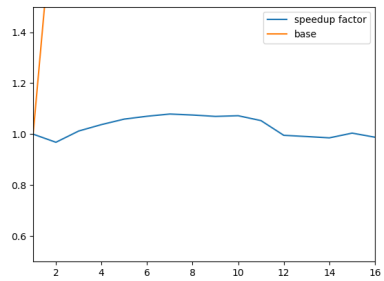


Рис. 6: Разные

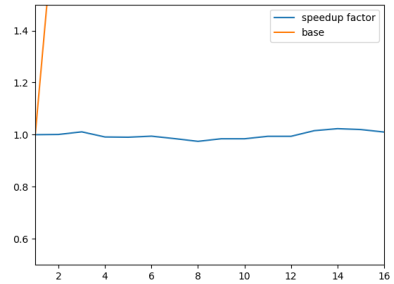


Рис. 7: Одинаковые

Эффективность

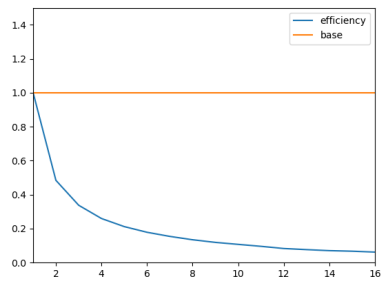


Рис. 8: Разные

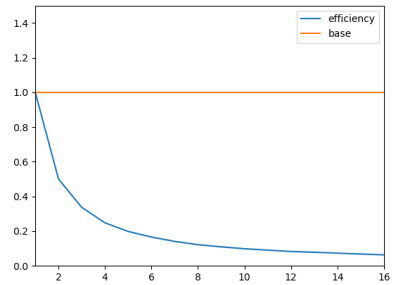


Рис. 9: Одинаковые

5 Приложение 1

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#include <string.h>
#include <float.h>

// #define __SAME_DATA__

int max_fn(int* array, const int size, const int threads_num)
{
    int max = array[0];
    #pragma omp parallel for num_threads(threads_num) reduction(max: max)
    for (int i = 0; i < size; ++i)
        max = array[i] > max ? array[i] : max;

    return max;
}

int main(int argc, char** argv)
{
    printf("OpenMP version: %d\n", _OPENMP);

    const int TESTS_NUM = 10;
    const int THREADS_NUM_MAX = 16;
    const int size = 1e8;
    int* array = (int*)malloc(size * sizeof(int));

    for (int i = 0; i < TESTS_NUM; ++i)
    {
        srand(time(NULL));

#ifndef __SAME_DATA__
        for (int j = 0; j < size; ++j)
            array[j] = rand();
#else
        memset(array, 0, size * sizeof(int));
#endif /* ! __SAME_DATA__ */

        double times[THREADS_NUM_MAX];
        for (int threads_num = 1; threads_num <= THREADS_NUM_MAX; ++threads_num)
        {
            double start = omp_get_wtime();
```

```
            max_fn(array, size, 5);
            double end = omp_get_wtime();

            times[threads_num - 1] = end - start;

            printf ("Test #%d threads_num: %2d time: %fs\n", i + 1, threads_num,
        }

        int pos = -1;
        double min = DBL_MAX;
        for (int j = 0; j < THREADS_NUM_MAX; ++j)
            if (times[j] < min)
            {
                min = times[j];
                pos = j;
            }

        printf("Best time: %f threads_num: %d\n", min, pos + 1);
    }

    free(array);

    return 0;
}
```

6 Приложение 1а

```
from os import write
import matplotlib.pyplot as plt
import re
from statistics import mean
import sys
import csv

INPUT_FILE = "log.txt"
REGEXP = r"Test \#(.*?) threads_num: (.*?) time: (.*?)s\n"
MAX_THREADS_NUM = 16
OUTPUT_PATH = "../images/"
EXT_FILENAME = ""

def write_csv(table):
    with open("data" + EXT_FILENAME + ".csv", "w", newline="") as csvfile:
        writer = csv.writer(csvfile)

        headers = ["Threads", "Avg"]
        tests = [j + 1 for j in range(10)]
        headers.extend(tests)
        writer.writerow(headers)

        for i in range(MAX_THREADS_NUM):
            tmp = [i + 1, mean(table[i])]
            tmp.extend(table[i])
            writer.writerow(tmp)

def load_table(input_filename, regexp, max_threads):
    with open(input_filename, "r") as input:
        content = input.read()

    data = re.findall(regexp, content)

    table = [[] for i in range(max_threads)]
    for str in data:
        test, threads, time = str
        table[int(threads) - 1].append(float(time))

    return table

def make_plots(x, y, y_theor, legend, f_name, x_min=1, x_max=MAX_THREADS_NUM, y_
    plt.axis([x_min, x_max, y_min, y_max])
    plt.plot(x, y)
```

```
    plt.plot(x, y_theor)
    plt.legend(legend)
    plt.savefig(OUTPUT_PATH + f_name + EXT_FILENAME + ".png")
    plt.clf()


if __name__ == "__main__":
    if len(sys.argv) == 2:
        EXT_FILENAME = sys.argv[1]

    table = load_table(INPUT_FILE, REGEXP, MAX_THREADS_NUM)

    time = [mean(table[i]) for i in range(MAX_THREADS_NUM)]
    time_theor = [time[0] / (i + 1) for i in range(MAX_THREADS_NUM)]
    x = [(i + 1) for i in range(MAX_THREADS_NUM)]
    make_plots(x, time, time_theor, ["time", "base"], "time", y_max=max(time))

    accel = [time[0] / time[i] for i in range(MAX_THREADS_NUM)]
    accel_theor = [time_theor[0] / time_theor[i] for i in range(MAX_THREADS_NUM)
    make_plots(x, accel, accel_theor, ["speedup␣factor", "base"], "accel", y_min

    eff = [accel[i] / (i + 1) for i in range (MAX_THREADS_NUM)]
    eff_theor = [accel_theor[i] / (i + 1) for i in range (MAX_THREADS_NUM)]
    make_plots(x, eff, eff_theor, ["efficiency", "base"], "eff")

    write_csv(table)
```

7 Приложение 1б

```
#!/bin/bash

echo "[RAND_DATA]"
echo "Compiling"
gcc -o lab1 -fopenmp lab1.c
echo "Working..."
./lab1 > log.txt
echo "Making plots"
python3 script.py

echo "[SAME_DATA]"
echo "Compiling"
gcc -o lab1 -D __SAME_DATA__ -fopenmp lab1.c
echo "Working..."
./lab1 > log.txt
echo "Making plots"
python3 script.py _same

# echo "Constructing report"
# latex ../report.tex
```

8 Приложение 2

Таблица 1: Разные

Threads	Avg	1	2	3	4	5	6	7	8	9	10
1	0.0580845	0.057363	0.073631	0.06308	0.061089	0.048023	0.069911	0.048871	0.059335	0.052136	0.047406
2	0.059997	0.056855	0.074048	0.05894	0.086836	0.048525	0.070037	0.048773	0.055398	0.052871	0.047687
3	0.0573792	0.057025	0.070381	0.057711	0.065674	0.048481	0.070121	0.049266	0.054983	0.052505	0.047645
4	0.0559819	0.057143	0.056262	0.057589	0.077522	0.048678	0.0594	0.048763	0.053999	0.052637	0.047826
5	0.0548461	0.057486	0.056409	0.057235	0.073867	0.048513	0.049123	0.049341	0.055239	0.053389	0.047859
6	0.0542699	0.057358	0.056359	0.057503	0.06796	0.048884	0.048855	0.049227	0.055578	0.053002	0.047973
7	0.053832200000000004	0.057717	0.056644	0.054497	0.064405	0.049322	0.049659	0.049493	0.055897	0.052637	0.048051
8	0.0540287	0.057351	0.056724	0.056982	0.064156	0.049033	0.049219	0.052367	0.053955	0.052614	0.047886
9	0.0542988	0.057177	0.056914	0.059721	0.065311	0.048982	0.04921	0.049514	0.055398	0.052469	0.048292
10	0.0541678	0.05717	0.056789	0.059417	0.064478	0.048787	0.049115	0.049354	0.05521	0.052812	0.048546
11	0.0551514	0.056888	0.059297	0.060999	0.067652	0.048722	0.049394	0.049083	0.055377	0.055995	0.048107
12	0.0583518	0.05722	0.056842	0.088709	0.066168	0.049514	0.049539	0.049087	0.053234	0.065137	0.048068
13	0.0586271	0.057043	0.056642	0.084135	0.07839	0.048736	0.049095	0.049502	0.050174	0.064624	0.04793
14	0.058943	0.05691	0.05666	0.076462	0.08739	0.048949	0.04923	0.049001	0.050538	0.064651	0.049639
15	0.057849399999999995	0.057203	0.056761	0.074775	0.081688	0.048723	0.049335	0.049803	0.049883	0.062533	0.04779
16	0.0587868	0.057226	0.056894	0.074329	0.067573	0.049288	0.049091	0.050592	0.077018	0.057522	0.048335

Таблица 2: Одинаковые

Threads	Avg	1	2	3	4	5	6	7	8	9	10
1	0.0652923	0.071391	0.049936	0.049483	0.067586	0.06596	0.070817	0.070624	0.068249	0.070604	0.068273
2	0.0652358	0.070183	0.049895	0.049838	0.067981	0.065903	0.070589	0.070961	0.068146	0.070618	0.068244
3	0.0645853	0.067707	0.051977	0.050128	0.067407	0.066402	0.068034	0.068178	0.069523	0.068232	0.068265
4	0.06587	0.076558	0.050428	0.052305	0.067898	0.068081	0.067968	0.068167	0.070905	0.068251	0.068139
5	0.0659299	0.074463	0.049875	0.053898	0.067614	0.068199	0.067943	0.067894	0.070989	0.068203	0.070221
6	0.0656645	0.069647	0.048877	0.055757	0.06759	0.067994	0.069168	0.067927	0.070303	0.068162	0.07122
7	0.0662951	0.072513	0.049112	0.057555	0.067629	0.068222	0.071059	0.069538	0.068272	0.067994	0.071057
8	0.0670065	0.066994	0.050342	0.068422	0.067646	0.068009	0.071021	0.071191	0.068313	0.068022	0.070105
9	0.0663286	0.061852	0.049838	0.068457	0.067548	0.068027	0.07104	0.071015	0.068153	0.069285	0.068071
10	0.0663306	0.061492	0.04943	0.067955	0.068227	0.068109	0.07082	0.070311	0.068024	0.070958	0.06798
11	0.065702	0.061926	0.049164	0.067551	0.0665	0.068311	0.068037	0.068144	0.068131	0.071008	0.068248
12	0.0657011	0.061515	0.049724	0.071215	0.062993	0.068287	0.067943	0.068175	0.068337	0.070774	0.068048
13	0.0643002	0.054713	0.049441	0.067025	0.063473	0.068023	0.06804	0.067847	0.068078	0.068274	0.068088
14	0.0638162999999999	0.049126	0.049371	0.06763	0.063597	0.067882	0.068178	0.068112	0.068241	0.068032	0.067994
15	0.0640249	0.049041	0.049136	0.067617	0.065013	0.069018	0.068085	0.068224	0.068015	0.068139	0.067961
16	0.0646306	0.049781	0.048955	0.068204	0.06564	0.070722	0.069413	0.068079	0.069721	0.067656	0.068135