

Параллельное программирование
Лабораторная работа №1

ИИКС ИБ
Б19-505
Голигузов Алексей
Сентябрь 2021

Содержание

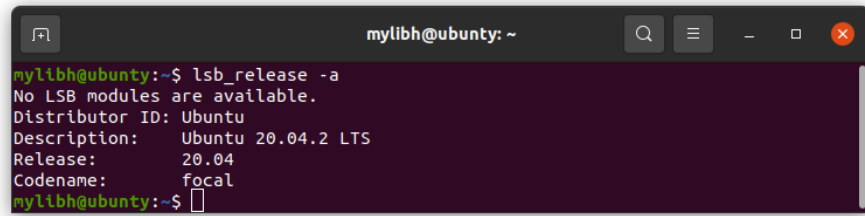
1	Рабочая среда	3
2	Анализ	5
3	Графики	6
4	Заключение	7
5	Приложение 1	8
6	Приложение 1а	10
7	Приложение 1б	12
8	Приложение 2	13

1 Рабочая среда

```
mylibh@ubuntu: ~  
mylibh@ubuntu:~$ lscpu  
Architecture:          x86_64  
CPU op-mode(s):        32-bit, 64-bit  
Byte Order:             Little Endian  
Address sizes:          39 bits physical, 48 bits virtual  
CPU(s):                 8  
On-line CPU(s) list:    0-7  
Thread(s) per core:     2  
Core(s) per socket:     4  
Socket(s):              1  
NUMA node(s):           1  
Vendor ID:              GenuineIntel  
CPU family:              6  
Model:                  142  
Model name:              Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz  
Stepping:                12  
CPU MHz:                 800.007  
CPU max MHz:             4900.0000  
CPU min MHz:             400.0000  
BogoMIPS:                4599.93  
Virtualization:          VT-x  
L1d cache:               128 KiB  
L1i cache:               128 KiB  
L2 cache:                1 MiB  
L3 cache:                8 MiB
```

```
mylibh@ubuntu: ~  
mylibh@ubuntu:~$ free -h  
              total        used        free      shared  buff/cache   available  
Mem:          15Gi        3.0Gi        9.2Gi        720Mi        3.2Gi        11Gi  
Swap:         29Gi         0B         29Gi  
mylibh@ubuntu:~$
```

```
mylibh@ubuntu: ~  
mylibh@ubuntu:~$ echo |cpp -fopenmp -dM |grep -i open  
#define _OPENMP 201511  
mylibh@ubuntu:~$ gcc --version  
gcc (Ubuntu 10.3.0-1ubuntu1~20.04) 10.3.0  
Copyright (C) 2020 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
mylibh@ubuntu:~$
```



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

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

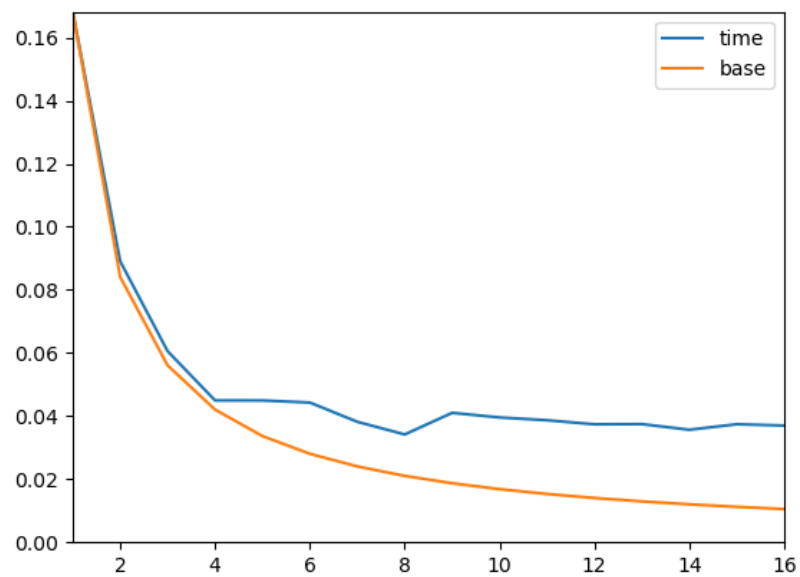


Рис. 1: Время

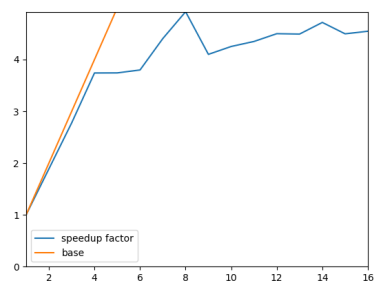


Рис. 2: Ускорение

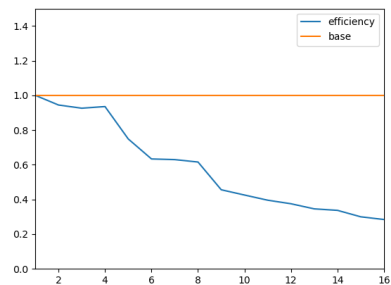


Рис. 3: Эффективность

4 Заключение

Время

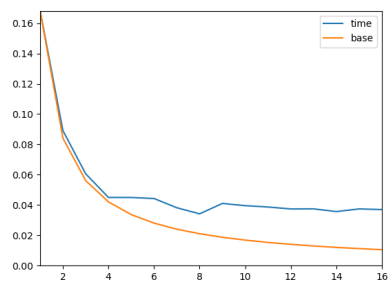


Рис. 4: Разные

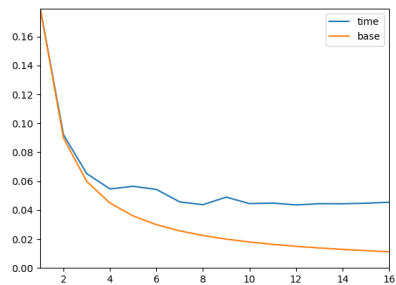


Рис. 5: Одинаковые

Ускорение

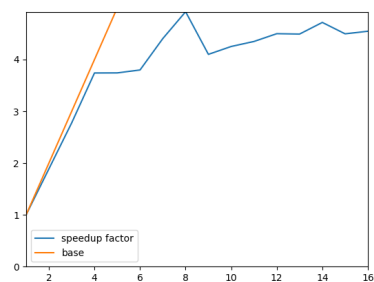


Рис. 6: Разные

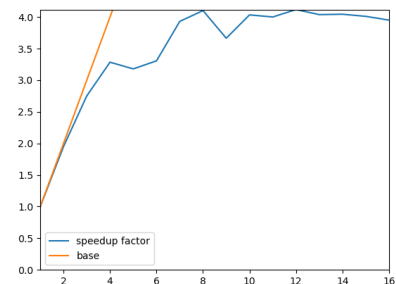


Рис. 7: Одинаковые

Эффективность

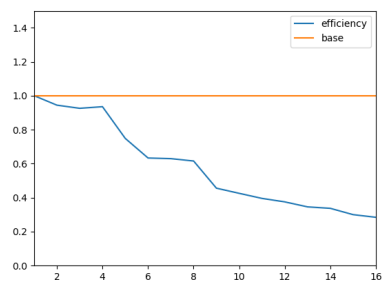


Рис. 8: Разные

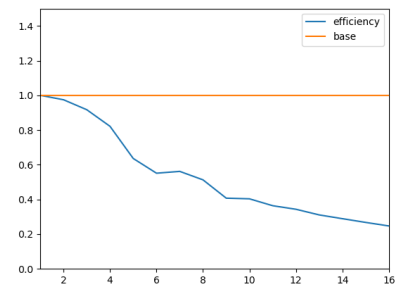


Рис. 9: Одинаковые

5 Приложение 1

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

// #define __SAME_DATA__
inline int max_fn(int* array, const int size, const int threads_num);
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));

#ifdef __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, threads_num);
    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 1a

```
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_min=0, y_max=10):
    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_max=

    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 16

```
#!/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	0.1682223	0.170904	0.168587	0.167573	0.166744	0.167908	0.16978	0.167524	0.168243	0.166334	0.168626
2	0.0890898	0.08591	0.099633	0.086343	0.084106	0.085818	0.08588	0.086021	0.085528	0.10017	0.091489
3	0.0605758	0.056962	0.076004	0.056498	0.056392	0.05657	0.05661	0.057335	0.056471	0.076432	0.056484
4	0.0449695	0.043044	0.051932	0.042367	0.042358	0.042418	0.042517	0.042958	0.042639	0.057094	0.042368
5	0.0449495	0.044699	0.045362	0.04505	0.044696	0.045286	0.044664	0.045783	0.044617	0.044612	0.044726
6	0.044279	0.044489	0.044112	0.043983	0.04404	0.044868	0.044024	0.045078	0.044055	0.044029	0.044112
7	0.038177	0.038319	0.037824	0.03799	0.038121	0.038891	0.037766	0.039101	0.037879	0.037729	0.03815
8	0.0341606	0.034044	0.035415	0.03413	0.033891	0.034723	0.033464	0.034942	0.033713	0.033467	0.033817
9	0.0410242	0.041313	0.040356	0.040658	0.040837	0.041927	0.040573	0.042174	0.040621	0.040713	0.04107
10	0.0395554	0.040072	0.036889	0.039863	0.040096	0.037682	0.040148	0.040278	0.039275	0.039365	0.041886
11	0.0386737	0.038369	0.03806	0.039793	0.044764	0.03821	0.038534	0.03893	0.036978	0.037401	0.035698
12	0.0373845	0.035957	0.044019	0.038876	0.03661	0.03873	0.034591	0.03697	0.034666	0.038128	0.035298
13	0.0374445	0.041562	0.036942	0.036595	0.040978	0.037039	0.035393	0.038959	0.035729	0.034412	0.036836
14	0.0356599	0.036856	0.033951	0.035065	0.034736	0.034898	0.036296	0.037412	0.036796	0.034533	0.036056
15	0.0374054	0.036604	0.033751	0.037359	0.035371	0.051688	0.037105	0.036126	0.035169	0.036793	0.034088
16	0.0369946	0.034057	0.033174	0.033964	0.038596	0.048378	0.034684	0.039867	0.033551	0.034133	0.039542

Таблица 2: Одинаковые

Threads	Avg	1	2	3	4	5	6	7	8	9	10
1	0.179454	0.21955	0.181585	0.171757	0.199274	0.17185	0.170909	0.17122	0.169257	0.170094	0.169044
2	0.0920782	0.12841	0.085688	0.090585	0.085577	0.085549	0.085539	0.086421	0.08692	0.089166	0.096927
3	0.0652273	0.092441	0.056482	0.063394	0.061093	0.061279	0.060587	0.065371	0.06514	0.057177	0.069309
4	0.0546211	0.091523	0.042582	0.052062	0.049593	0.049728	0.049411	0.053699	0.053588	0.046964	0.057061
5	0.056443	0.086778	0.045045	0.054343	0.052039	0.052045	0.051974	0.055956	0.056658	0.04964	0.059952
6	0.0542722	0.072006	0.044214	0.053485	0.051284	0.051357	0.051484	0.055529	0.05608	0.048452	0.058831
7	0.0456508	0.045509	0.038239	0.046974	0.044799	0.044791	0.044744	0.048511	0.048498	0.042166	0.052277
8	0.0437464	0.066136	0.034096	0.042211	0.040528	0.040762	0.040463	0.043756	0.044014	0.038221	0.047277
9	0.0489646	0.048516	0.043187	0.050398	0.046789	0.047803	0.04755	0.053307	0.051081	0.045391	0.055624
10	0.0444879	0.046998	0.041775	0.046458	0.044581	0.042755	0.043009	0.049553	0.040068	0.040593	0.049089
11	0.0448555	0.043515	0.041689	0.050176	0.046128	0.046818	0.042824	0.046987	0.04032	0.041248	0.04885
12	0.0435752	0.046727	0.035651	0.045237	0.04177	0.040807	0.041418	0.048326	0.042258	0.042507	0.051051
13	0.0444356	0.054273	0.040362	0.044415	0.041706	0.044568	0.04363	0.045015	0.040489	0.039604	0.050294
14	0.0443693	0.05797	0.0352	0.044569	0.041355	0.041352	0.043049	0.048334	0.041016	0.039634	0.051214
15	0.0447421	0.052424	0.036216	0.055627	0.04205	0.044073	0.044431	0.044497	0.039664	0.039615	0.048824
16	0.0454137	0.049117	0.041724	0.065304	0.043282	0.041513	0.040348	0.044969	0.041398	0.037957	0.048525