Отчёт по лабораторной работе №6

Окорочкова Мария, M32341

30 декабря 2022 г.

1. Детали реализации каждого алгоритма.

Алгоритм эксперимента № 1

```
#!/bin/bash

for (( i=0; i<1001; i++ )); do
        for (( j=0; j<501; j++ )); do
                result=$(($j+$i+$j*$i))
        done
done
```

Алгоритм эксперимента № 2

```
#!/bin/bash

file=$1

for (( i=0; i<1000; i++ )); do
        read num < $file
        num=$num$((2 * $num))
        echo $num > $file
done
```

Общий скрипт запуска

```
#!/bin/bash

experiment=$1
mode=$2
processors=$3
num=$4

path=logs/$experiment
algo=$path/algo

case $experiment in
        "1experiment" )
                case $mode in
                "seq" )
                        for (( i=0; i<$num; i++ )); do
                                bash $algo
                        done
                        ;;
                "parallel" )
                        for (( i=0; i<$num; i++ )); do
                                bash $algo &
                        done

                        ./pause
                        ;;
                esac
                ;;
        "2experiment" )
                for (( i=0; i<$num; i++ ));  do
                        path2=$path/$processors/$mode/files/file$i
                        : > $path2

                        theNumber=1
                        echo $theNumber$theNumber$theNumber >> $path2

                        case $mode in
                        "seq" )
                                bash $algo "$path2"
                                ;;
                        "parallel" )
                                bash $algo "$path2" &
                                ;;
                        esac
                done
                if [ "$mode" == "parallel" ]; then
                        ./pause
                fi
                ;;
esac

```

Общий скрипт оценки времени алгоритма (получаем при вызове time -p **значение параметра** real**)**

```
#!/bin/bash

experiment=$1
mode=$2
processors=$3

pathTime=logs/$experiment/$processors/$mode/time.log
pathReal=logs/$experiment/$processors/$mode/real.log

for (( i=1; i<21; i++ )); do
        exec 2>$pathTime
        for (( j=0; j<5; j++ )); do
                time -p ./launch $experiment $mode $processors $i
        done
        echo $(grep "real" $pathTime | awk '{print $2}') >> $pathReal
        echo "" > $pathTime
done

rm $pathTime
```

2. Параметры.

Хостовый компютер

Память	16 ГБ 2133 MHz LPDDR3
Имя процессора	4-ядерный процессор Intel Core i7
Скорость процессора	2,8 GHz
Количество процессоров	1
Общее количество ядер	4
Кэш 2-го уровня (в каждом ядре)	256 КБ
Кэш 3-го уровня	6 МБ

Виртуальная машина

Основная память	2048 МБ
Порядок загрузки	Гибкий лиск -> Оптический диск -> Жёсткий диск
Чипсет	PIIX3 TPM
отсутствует Процессор	1 или 2 ЦП
Предел загрузки ЦПУ	100%

3. Дополненные планы экспериментов.

Запускаем скрипты и берём логи с помощью следующих команд

```
./range 1experiment seq 1
./range 1experiment parallel 1
./range 1experiment seq 2
./range 1experiment parallel 2
./range 2experiment seq 1
```

```
./range 2experiment parallel 1
./range 2experiment seq 2
./range 2experiment parallel 2,
```

где первый параметр – эксперимент, второй – способ запуска, третий – кол-во процессоров.

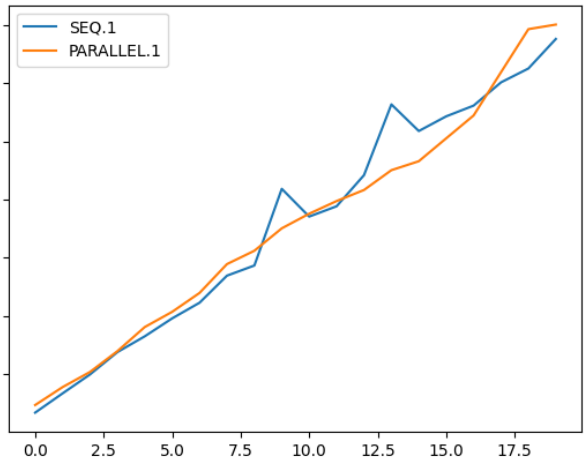
К сожалению, пришлось не соблюсти условие о количестве запусков скрипта (в цикле от 1 до 20) для подсчёта среднего арифметичееского выполнения алгоритма. Дело в том, что, если запускать, как предлагается, 10 раз, то придётся потратить на выполнение работы 13-14 часов, что затруднительно. Поэтому приняла решение запускать скрипт не 10 раз, а 5.

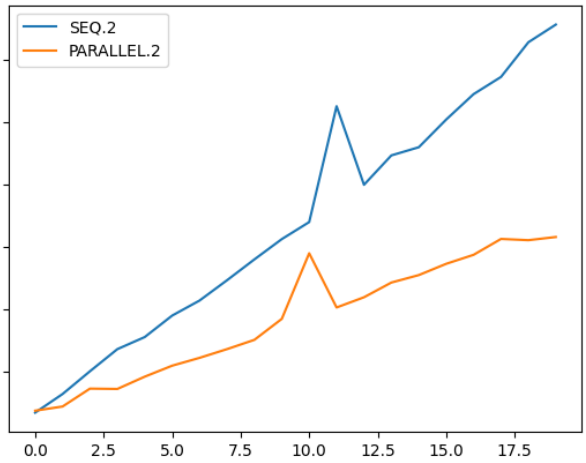
4. Графики + комментарии.

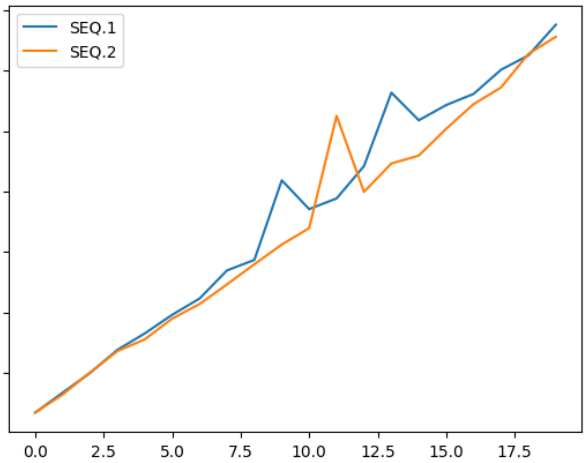
Искомое среднее время выполнения:

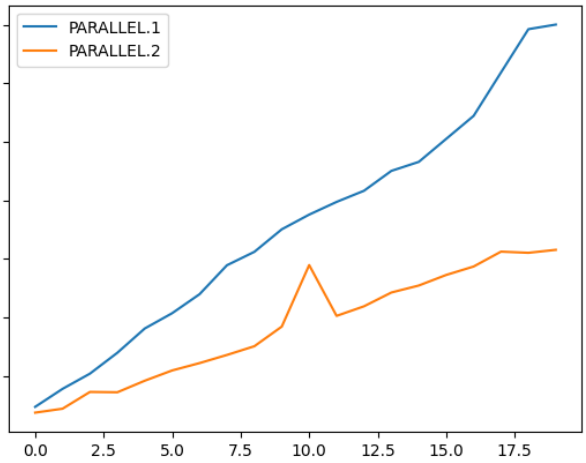
EXPERIMENTS							
1CPU		2CPU		1CPU		2CPU	
seq	parallel	seq	parallel	seq	parallel	seq	parallel
3.400	4.708	3.406	3.716	1.694	2.220	1.680	1.748
6.704	7.784	6.390	4.390	2.846	4.410	3.218	2.166
9.956	10.388	10.0380	7.258	4.992	5.106	4.880	3.340
13.796	13.958	13.584	7.202	5.556	6.618	7.108	4.250
16.514	18.106	15.524	9.174	6.740	9.628	6.948	3.930
19.596	20.722	18.964	10.932	8.284	8.952	8.976	4.612
22.288	23.972	21.376	12.190	10.800	12.102	9.326	6.464
26.936	28.924	24.622	13.584	10.416	14.224	10.374	6.530
28.682	31.216	27.966	15.074	12.338	13.486	12.708	7.378
41.854	35.068	31.218	18.432	13.866	15.840	13.95	8.246
37.086	37.58	33.948	28.950	14.054	15.978	15.634	9.122
38.858	39.752	52.518	20.262	15.25	17.644	16.546	9.954
44.204	41.656	39.928	21.900	28.154	20.656	18.746	10.790
56.362	45.06	44.64	24.260	17.724	24.238	22.796	11.620
51.786	46.598	45.946	25.454	19.014	24.128	22.794	12.864
54.31	50.532	50.386	27.262	26.996	27.210	24.742	13.156
56.152	54.46	54.46	28.704	25.824	28.856	26.932	13.838
60.146	61.892	57.216	31.248	26.028	28.166	27.364	14.972
62.532	69.29	62.776	31.048	27.810	30.952	29.596	14.238
67.602	70.076	65.606	31.558	27.648	45.974	32.092	14.95

Эксперимент №1

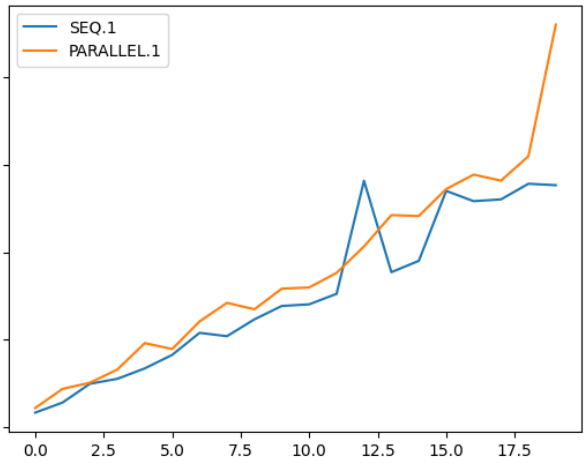
- 

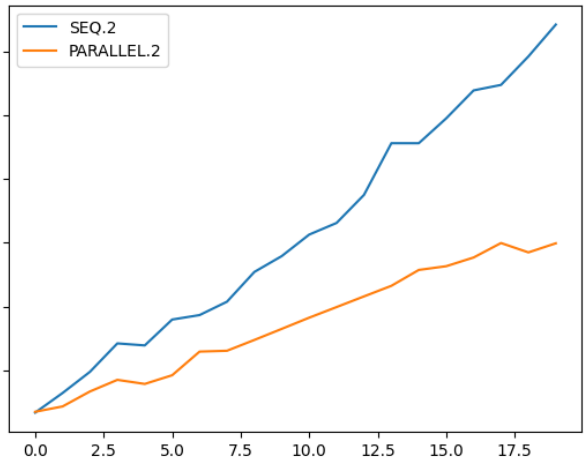


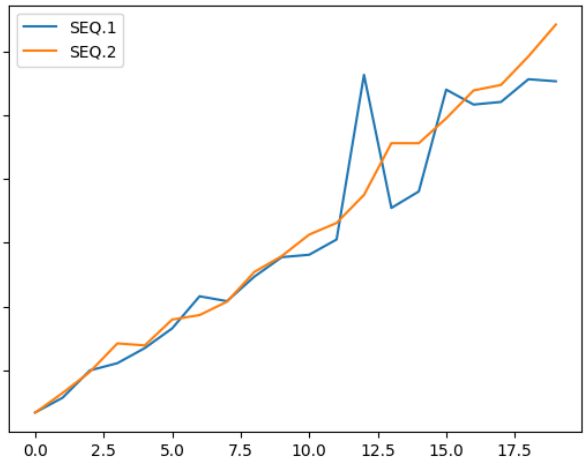
- 

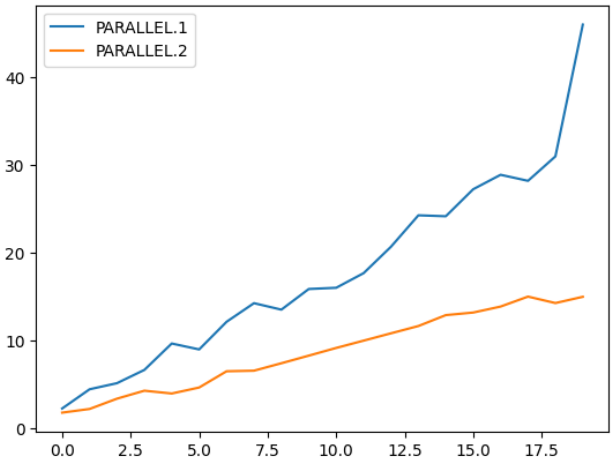


Эксперимент №2

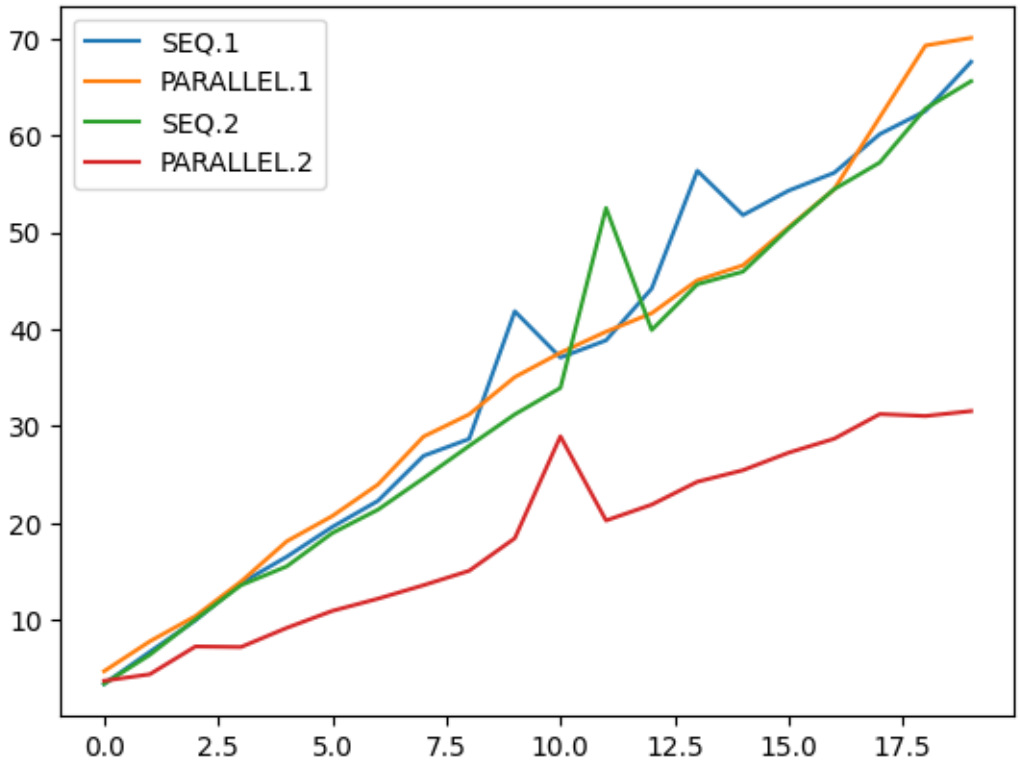
- 

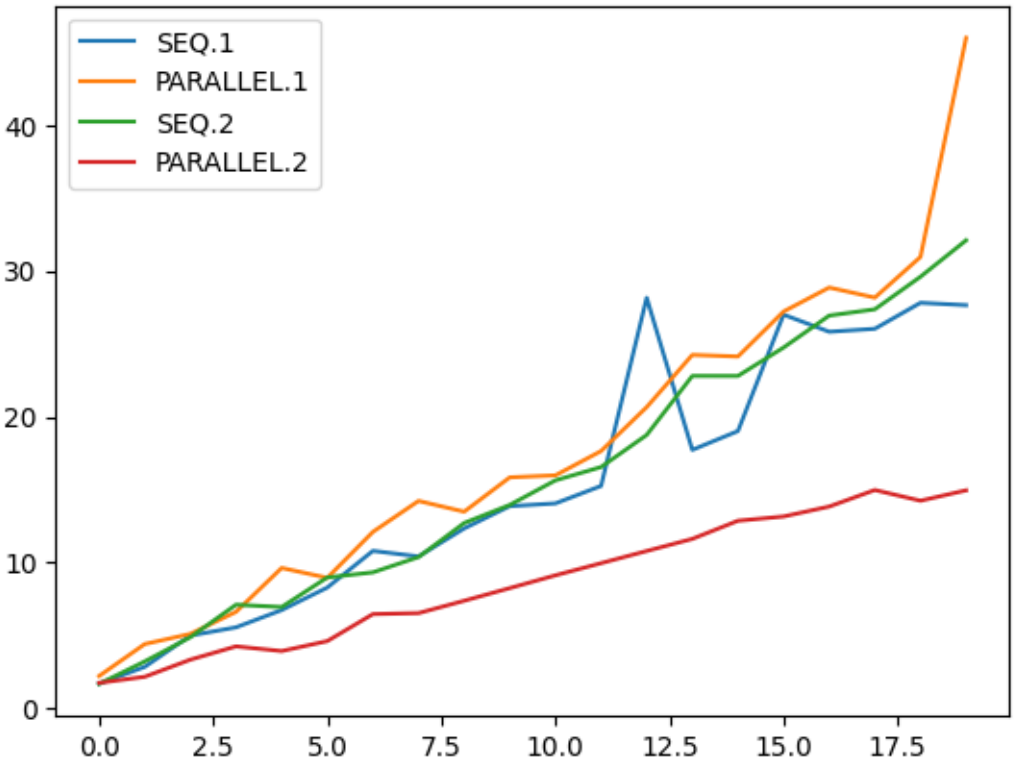



SEQ.1
SEQ.2
30
25
20
15
10
5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


PARALLEL.1
PARALLEL.2
40
30
20
10
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

-


SEQ.1
PARALLEL.1
SEQ.2
PARALLEL.2
70
60
50
40
30
20
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



Комментарии:

- Один процессор, сравнение последовательного и параллельного вызовов: значительной разницы по времени выполнения не наблюдается.
- Два процессора, сравнение последовательного и параллельного вызовов: выражается более быстрое выполнение при параллельном вызове.
- Сравнение последовательных вызовов при разном кол-ве процессоров: значительной разницы по времени выполнения не наблюдается.
- Сравнение параллельных вызовов при разном кол-ве процессоров: выражается более быстрое выполнение при двух процессорах.
- Больше всего времени тратится на параллельные вызовы скрипта при одном потоке, а меньше всего – на параллельные вызовы скрипта при двух потоках.
- Характер поведения времени первого и второго экспериментов крайне схож.
- Пристуствует некоторая погрешность ("скачки"на графиках).

5. Выводы:

- Эффективнее, конечно, использовать два потока(процессора), ведь тогда нагрузка будет распределяться равномерно.
- Параллельный способ запусков следует использовать при выполнении скрипта на двух процессорах/потоках, так быстрее. Опять же, играет роль распредление нагрузки.

- Последовательный способ запусков следует использовать при выполнении скрипта на одном процессоре/потоке, так быстрее. Параллельность на одном процессоре будет только мешать.
- Эксперимент №1 с выполнением вычислительно сложных задач выполняется медленее эксперимента № 2 с выполнением задач с большими объемами считываемых и сохраняемых данных, что ожидаемо.