Лабораторная работа №6

Сначала зафиксируем параметры хостового компьютера(лабораторная работа выполнялась на wsl2):

```
MiB Mem :   7837.2 total,   7166.0 free,    499.8 used,    171.5 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7122.2 avail Mem
```

```
galking@DESKTOP-64HAE7K:~/lab6$ 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):                          2
On-line CPU(s) list:             0,1
Thread(s) per core:              2
Core(s) per socket:              1
Socket(s):                       1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           140
Model name:                      11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz
Stepping:                        2
CPU MHz:                         3187.201
BogoMIPS:                        6374.40
Hypervisor vendor:               Microsoft
Virtualization type:             full
L1d cache:                       48 KiB
L1i cache:                       32 KiB
L2 cache:                        1.3 MiB
L3 cache:                        8 MiB
```

Первый эксперимент

Для первого эксперимента был написан скрипт ex.sh, который вычисляет функцию e^x с помощью ряда Маклорена. Точность до 1000 члена была подобрана, чтобы достичь времени выполнения около 2 секунд независимо от x.

```bash
#!/bin/bash

x=$1
N=1000

result=1
term=1

for ((n=1; n<N; n++)); do
    term=$(echo "scale=9; $term*$x/$n" | bc)
    result=$(echo "scale=9; $result+$term" | bc)
done

echo "$result"
```

Для организации слежения были написаны скрипты track1.sh, track2.sh, track3.sh, track4.sh. Которые соответствуют последовательному и параллельному исполнению на одном процессоре, и последовательному и параллельному исполнению на двух процессорах. Для последовательных запусков использовался скрипт 1b.sh, для параллельных 2b.sh.

```bash
#!/bin/bash

for ((i = 1; i <= $1; i++)); do
  ./ex.sh $i
done
```

```bash
#!/bin/bash

for ((i = 1; i <= $1; i++)); do
  ./ex.sh $i &
done
wait
```

Скрипты 1b.sh, 2b.sh.

```bash
log="1.log"

if [[ -f $log ]]
then
    rm $log
fi

for ((n = 1; n <= 20; n++))
do
    echo "$n:" >> $log
    for ((k = 0; k < 10; k++))
    do
        \time -f "%e" ./1b.sh $n 2>>$log
    done
done
```

Следящий скрипт на примере track1.sh.

После проведения всех этапов эксперимента получаем график:

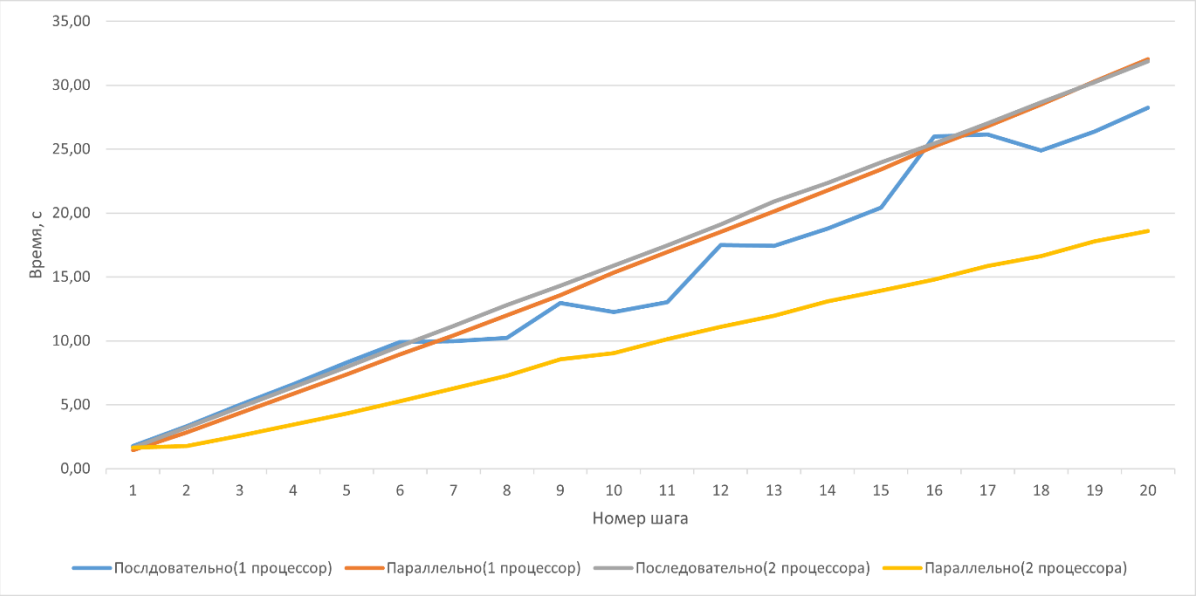

График зависимости времени работы от количества запусков.

Можем сделать выводы:

1. Последовательные запуски отработали примерно одинаково, можно грубо оценить линейной функцией. Это ожидаемо, наличие несколько вычислителей не играет роли, когда процессы запускаются последовательно, друг за другом.
2. Параллельный запуск на одном процессоре не дал никакого выигрыша по времени, отработал так же, как и последовательные запуски. Очевидно, что при наличии одного вычислителя параллельные запуски не работают ожидаемым образом и на самом деле процессы исполняются по очереди
3. Наконец, параллельный запуск на двух процессорах. График всё равно приближён к линейному, но время работы примерно в 2 раза меньше, чем в предыдущих запусках. Теперь планировщик может распределять по одному скрипту на процессор и заканчивать с ними работу примерно в одно и то же время, поэтому получаем выигрыш по общему времени исполнения.

Второй эксперимент

Для второго задания был написан генератор, который записывает в файлы числа от 0 до 9. Количество таких чисел в файле – 200_000. Таким образом, вес файлов равен 0,4 МБ. Такой размер был выбран для того, чтобы файл обрабатывался нужное по тз время. В моём случае на обработку одного файла уходило приблизительно 2.3 секунды.

```bash
#!/bin/bash

cnt=$(<.size)

for ((i=1; i <= $1; i++)); do
    name="files/file$i"
    echo -n "" > "$name"
    for ((j=1; j<=$cnt; j++)); do
        echo -en "$(($j % 10))\n" >> "$name"
    done
done
```

Скрипт gen.sh

Далее была написана программа, которая изменяет файл в соответствие с заданием.

```bash
#!/bin/bash

size=$1
name="files/file$2"
count=0
while read -r tmp && [ $count -lt $size ]; do
    echo -en "$((tmp * 2))\n" >> $name
    ((count++))
done < $name
```

Скрипт proc.sh

Скрипты для отслеживания практически в неизменном виде остались с первого эксперимента. Названия соответствуют описанию которое давалось в первом эксперименте.

В итоге, получаем график:

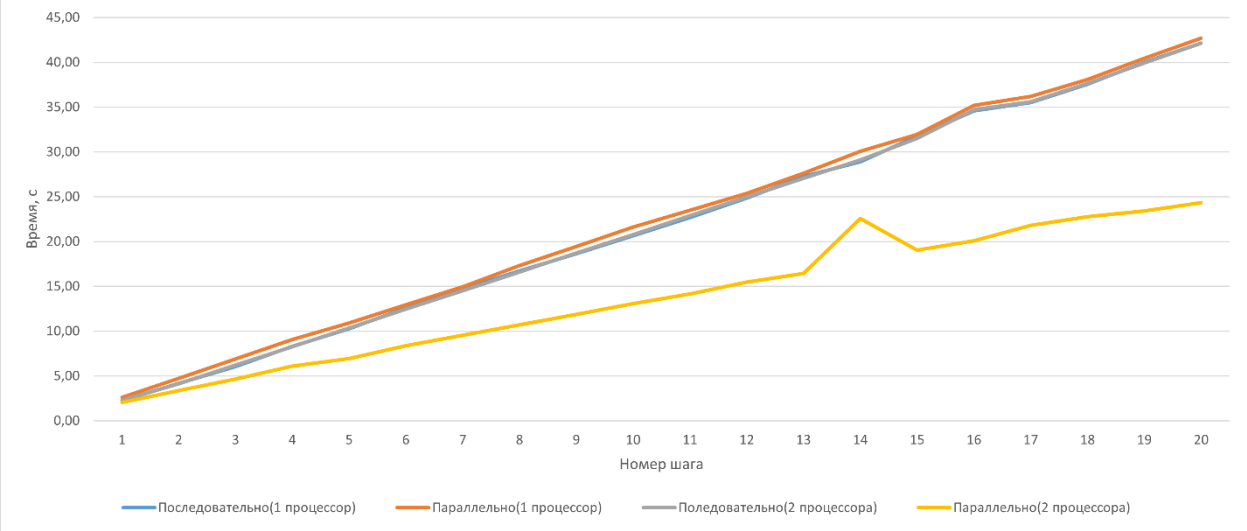

График зависимости времени работы от количества запусков.

Можем сделать выводы:

Результат соответствует полученным данным в первом эксперименте. Первые три запуска отработали одинаково, отличился высокой скоростью лишь параллельный запуск на двух процессорах.

Вывод

Исходя из результатов экспериментов можно сказать, что при наличии одного процессора, параллельные и последовательные вычисления занимают одинаковое количество времени. Более эффективной работы

можно добиться параллельными запусками на двух и более процессорах, время сокращается в 2 раза. Данные выводы актуальны и для работы с файлами, и для сложных вычислений.

Все скрипты и логи экспериментов прикладываю, с ними можно ознакомиться в репозитории.