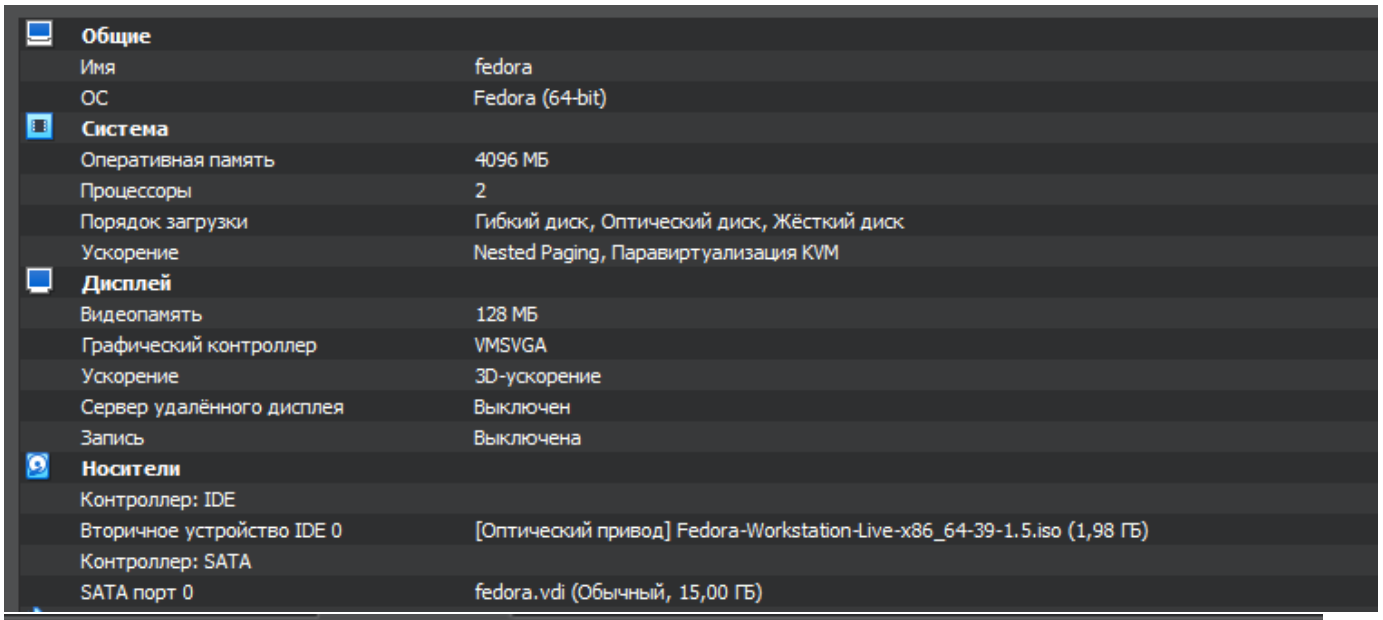
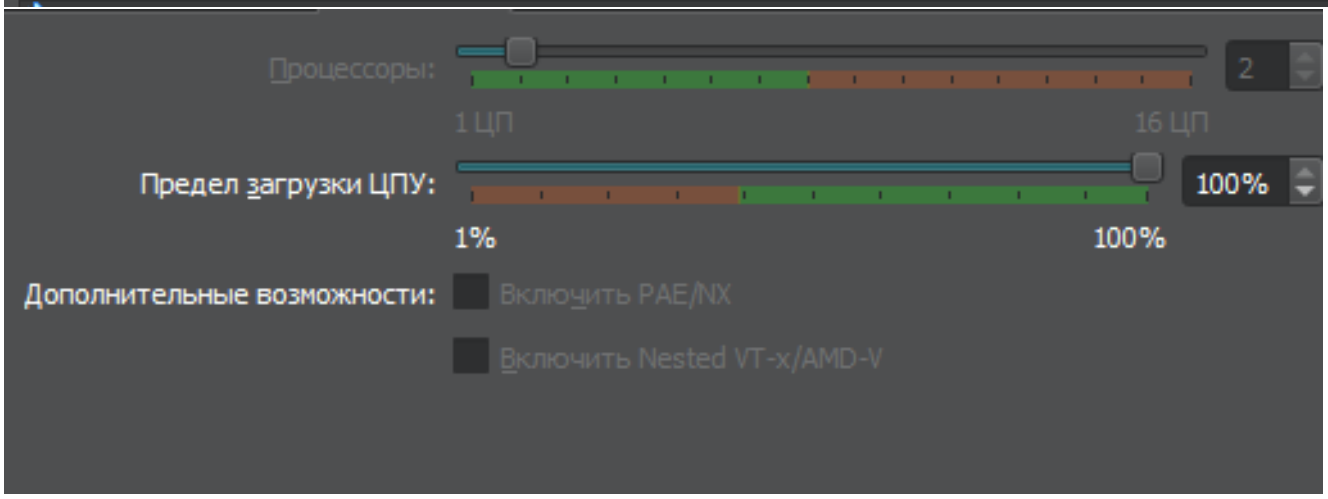
**Лабораторная работа №6 (вариант повышенной сложности). Экспериментальная оценка параметров производительности операционной системы**

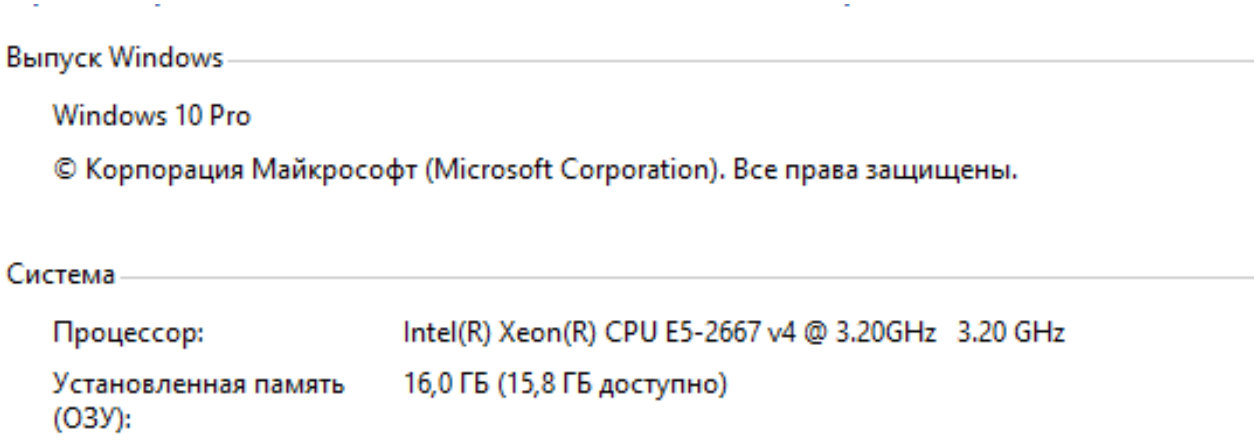
Цель лабораторной работы: оценить реальные накладные расходы на параллельное выполнение задач в условиях преимущественного использования ресурса процессора или ресурса дисковой подсистемы.

Параметры виртуальной машины:




Параметры основной системы:




Первый эксперимент

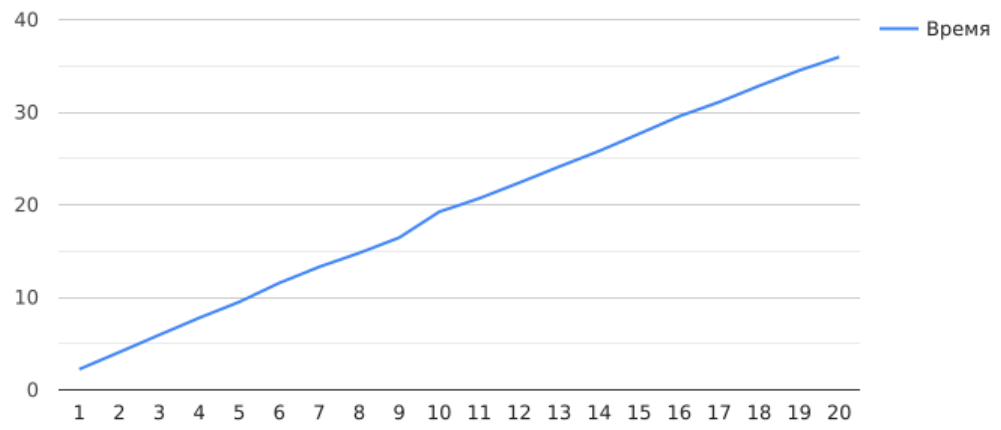
Основной скрипт выводит количество чисел кратных k до n=600000 O(n). Работает на одном числе примерно за 2 - 2.3 секунды.

```
#!/bin/bash

num=600000
for (( i = 1; i <= num; i++ )); do
    if (( i % $1 == 0 )); then
        (( count++ ))
    fi
done

echo "Result $1: $count"
```

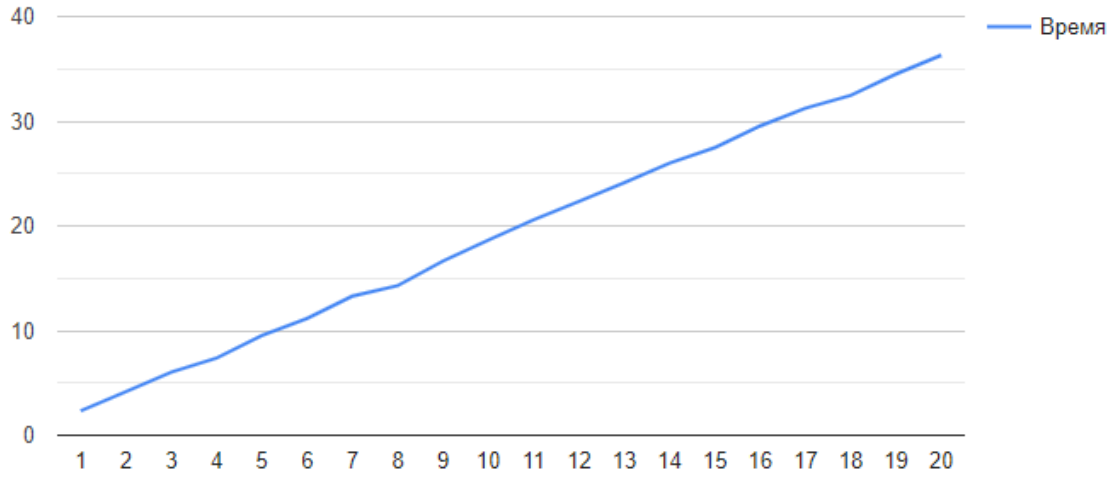
Наблюдатель:

```
#!/bin/bash

target=$1
logfile=$2

echo "" > $logfile

for ((i = 1; i <= 20; i++)); do
    count=0
    echo "n: $i" >> $logfile
    echo "--------------------" >> $logfile
    for ((j = 0; j < 1; j++)); do
        { \time -f "%E" ./$target $i ; } 2>> $logfile
    done
done
```

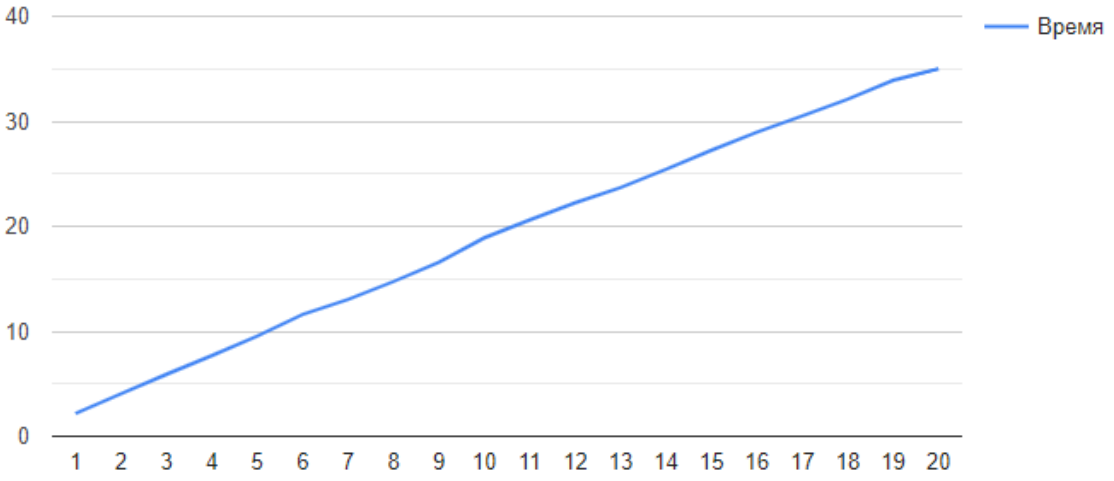
Графики:



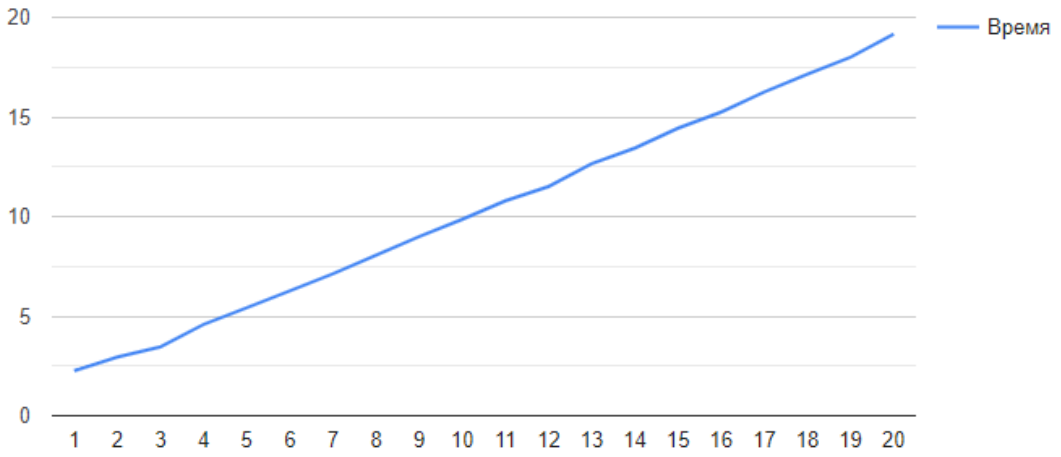

Последовательно на 1 процессоре




Параллельно на 1 процессоре




Последовательно на 2 процессорах




Параллельно на 2 процессорах

Обработка результатов:

1) В данном случае график примерно линейный. Все потому, что скрипты запускаются последовательно и исполняются примерно одинаково по времени.
2) Здесь происходит все так же, как и в первом случае потому, что скрипт исполняются на одном процессоре.
3) Функция так же примерно линейна. Хоть процессоров стала 2, однако скрипты запускаются последовательно.
4) В данном случая график все еще примерно линейный. Однако коэффициент уменьшился в примерно 2 раза (чуть меньше). Система распределяет скрипты по одному на процессор, и так потихоньку разгребается вся очередь.

## Второй эксперемент

Основной скрипт:

```bash
#!/bin/bash

input_file="files/$1.log"
cnt=0

while IFS= read -r value
do
    mul=$((value * 2))
    echo "$mul" >> "$input_file"
    (( cnt++ ))
    if [[ $cnt == 100000 ]]; then
        exit 0
    fi
done < "$input_file"
```

Скрипт для создания файлов:

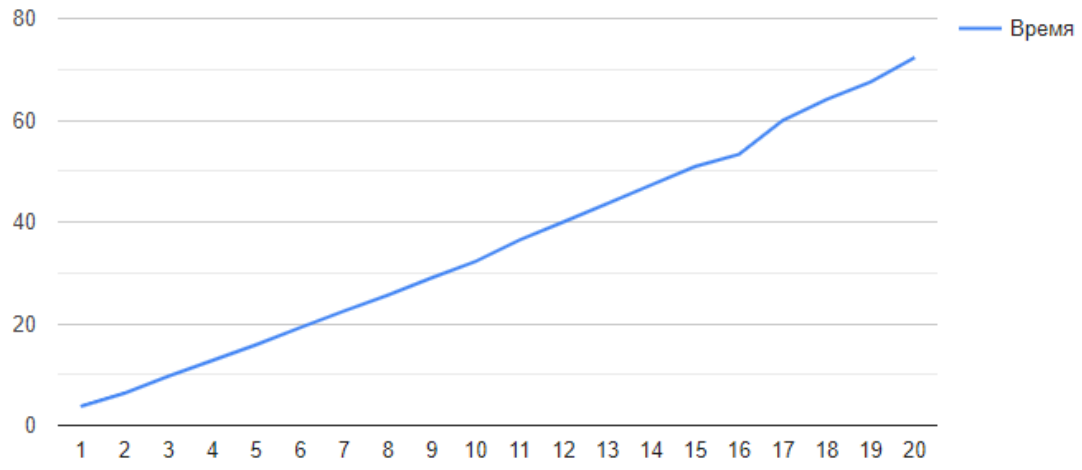
```bash
#!/bin/bash

size=100000
for ((i = 1; i <= $1; i++ )); do
    echo "1" > "files/$i.log"
    for ((j = 2; j <= size; j++)); do
        echo "$j" >> "files/$i.log"
    done
done
```

Наблюдатель:

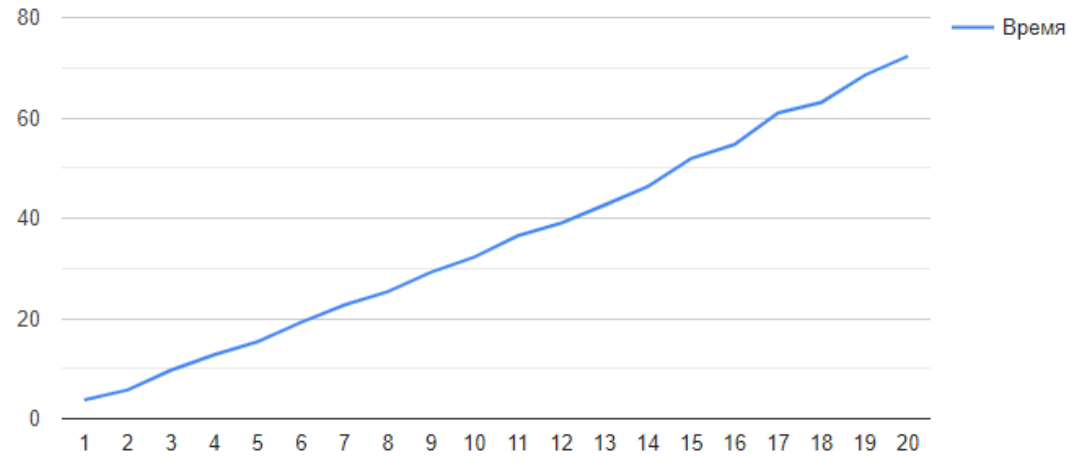
```
#!/bin/bash

target=$1
logfile=$2

echo "" > $logfile

for ((i = 1; i <= 20; i++)); do
    count=0
    echo "n: $i" >> $logfile
    echo "--------------------" >> $logfile
    for ((j = 0; j < 1; j++)); do
        ./make_files.sh $i
        { \time -f "%E" ./$target $i ; } 2>> $logfile
    done
done
```

Как можно заметить размер файлов - 100000. Время обработки чуть больше 3 секунд.

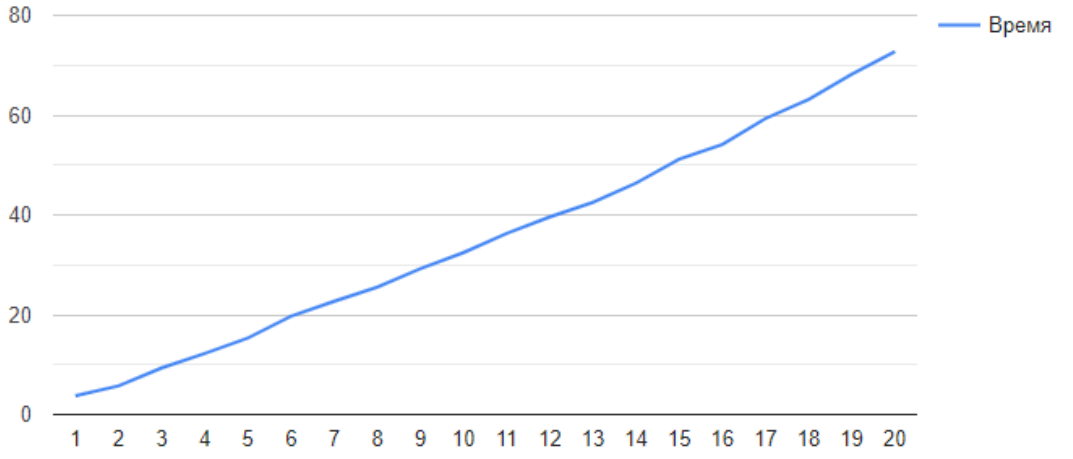
Графики:



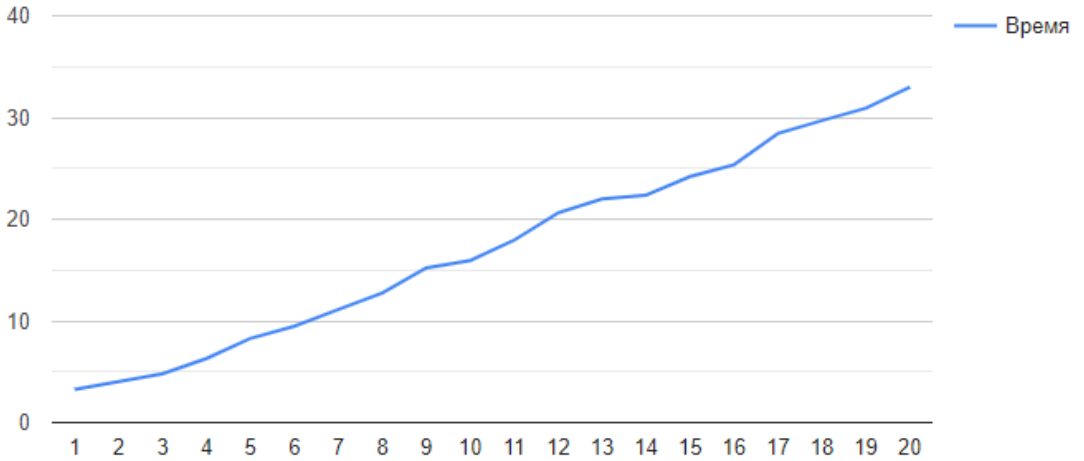

Последовательно на 1 процессоре




Параллельно на 1 процессоре




Последовательно на 2 процессорах



Параллельно на 2 процессорах

Обработка результатов:

Как и в предыдущем эксперименте, все графики примерно линейны. Последний график отличается коэффициентом.

Пропущенные скрипты:

```
#!/bin/bash

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

```
#!/bin/bash

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

wait
```

Эти 2 скрипта просто запускают основной скрипт согласно условию. Они совпадают для обоих экспериментов.