

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:

The screenshot displays the configuration interface for a virtual machine. It is divided into two main sections: a list of configuration categories on the left and a detailed settings table on the right. Below the table, there are interactive sliders for CPU settings and checkboxes for advanced features.

Категория	Параметр	Значение
Общие	Имя	fedora
	ОС	Fedora (64-bit)
Система	Оперативная память	4096 МБ
	Процессоры	2
	Порядок загрузки	Гибкий диск, Оптический диск, Жёсткий диск
	Ускорение	Nested Paging, Паравиртуализация KVM
Дисплей	Видеопамять	128 МБ
	Графический контроллер	VMSVGA
	Ускорение	3D-ускорение
	Сервер удалённого дисплея	Выключен
	Запись	Выключена
Носители	Контроллер: IDE	
	Вторичное устройство IDE 0	[Оптический привод] Fedora-Workstation-Live-x86_64-39-1.5.iso (1,98 ГБ)
	Контроллер: SATA	
	SATA порт 0	fedora.vdi (Обычный, 15,00 ГБ)

Процессоры: 2 (1 ЦП до 16 ЦП)

Предел загрузки ЦПУ: 100% (1% до 100%)

Дополнительные возможности:

- ☐ Включить PAE/NX
- ☐ Включить Nested VT-x/AMD-V

Параметры основной системы:

Выпуск Windows

Windows 10 Pro

© Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Система

Процессор:	Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz 3.20 GHz
Установленная память (ОЗУ):	16,0 ГБ (15,8 ГБ доступно)

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:

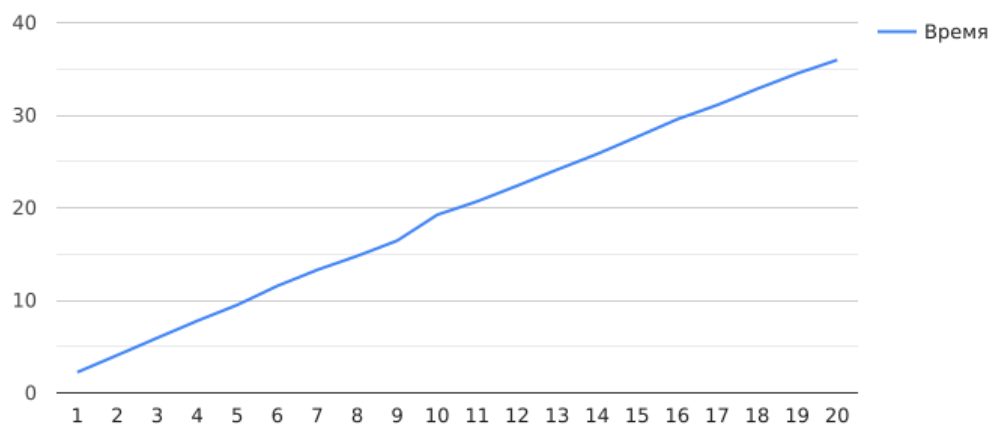
```
#!/bin/bash

target=$1
logfile=$2

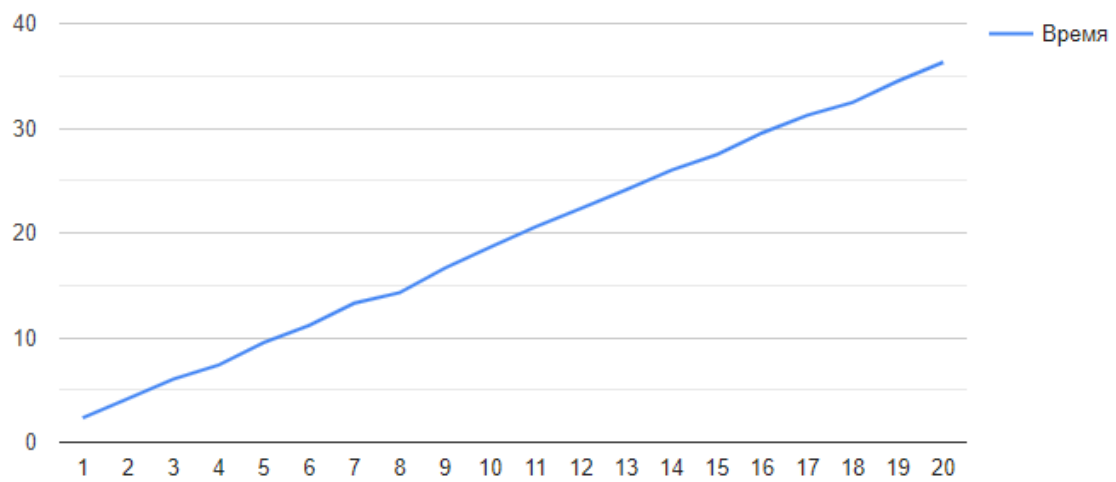
echo "" > $logfile

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
```

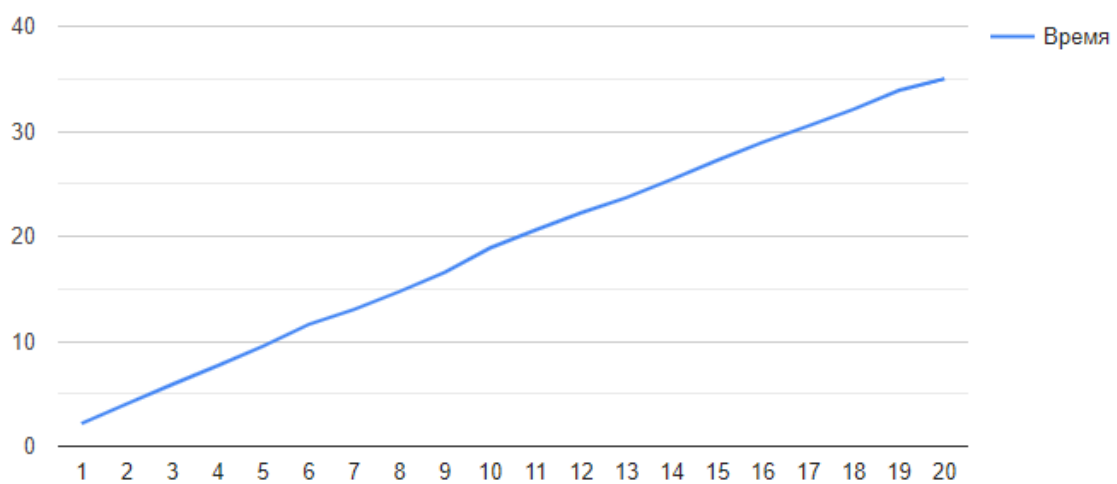
Графики:



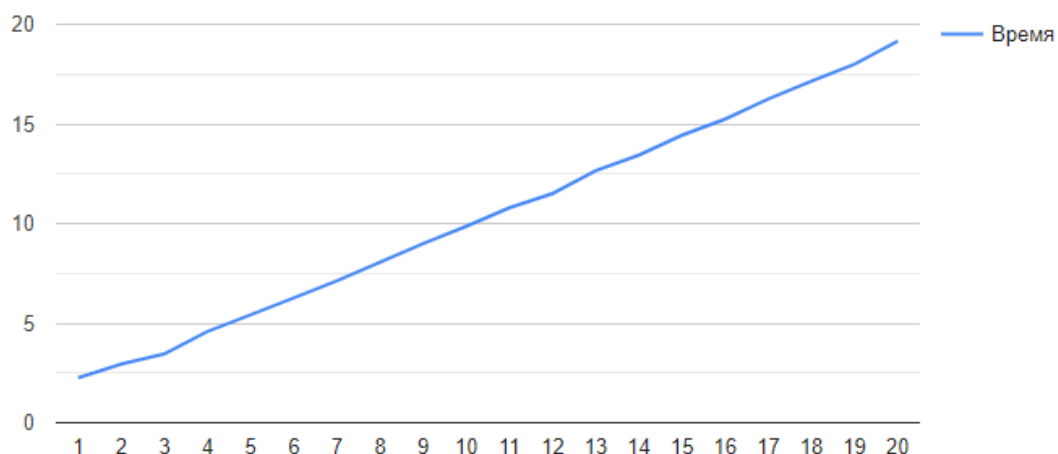
Последовательно на 1 процессоре



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 исполняется на одном процессоре.
- 3) Функция так же примерно линейна. Хотя процессоров стала 2, однако скрипты запускаются последовательно.
- 4) В данном случае 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 эксперимент

Основной скрипт:

```
#!/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:

```
#!/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

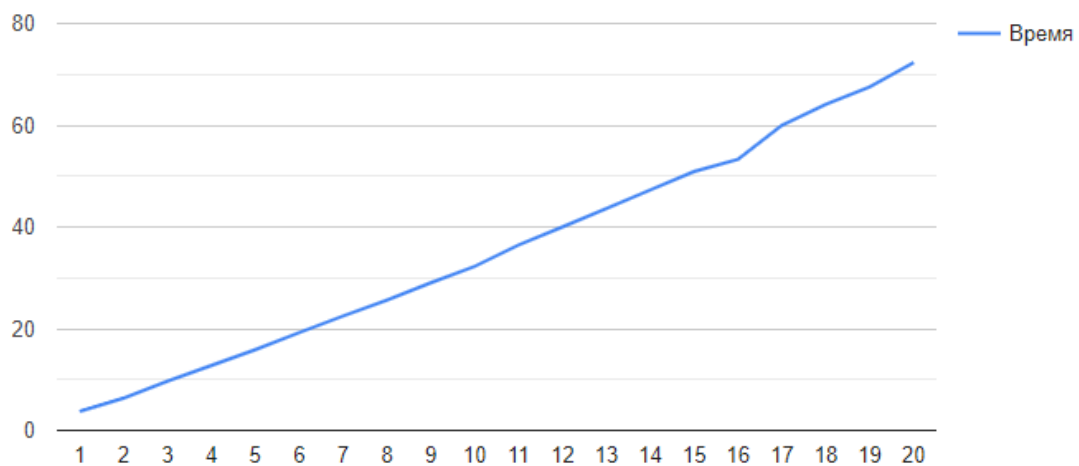
target=$1
logfile=$2

echo "" > $logfile

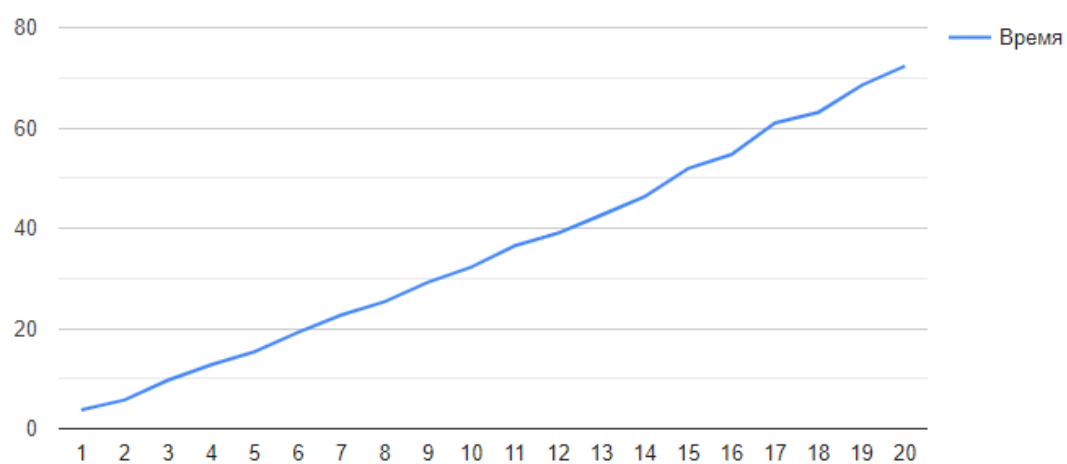
for ((i = 1; i <= 20; i++)); do
    count=0
    echo "n: $i" >> $logfile
    echo "-----" >> $logfile
    for ((j = 0; j < 1; j++)); do
        ./make_files.sh $i
        { \time -f "%E" ./$target $i ; } 2>> $logfile
    done
done
```

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

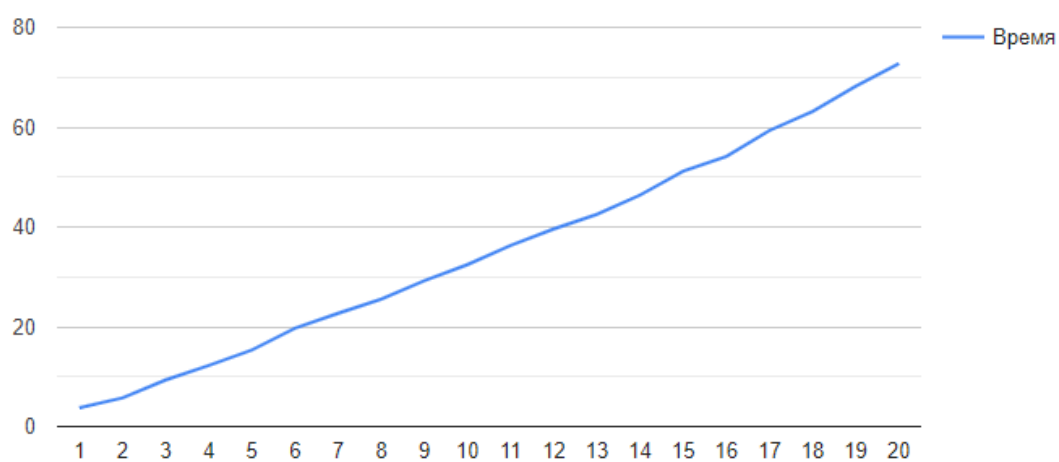
Графики:



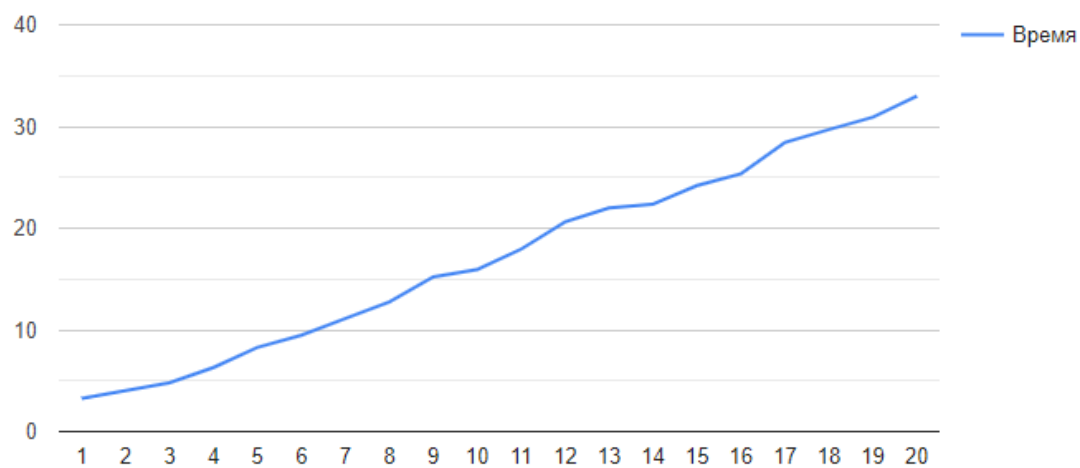
Последовательно на 1 процессоре



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.