

Федеральное государственное автономное
образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и
программирования

Операционные системы

Лабораторная работа №6

Выполнил: Гаджиев Саид М3215

Преподаватель: Дюкарева Вероника Максимовна

Санкт-Петербург

2024

Часть 1. Эксперименты с последовательным и параллельным выполнением вычислительно сложных задач

1. Первый эксперимент.

- a. В виртуальной машине настроить использование одного процессора.
- b. Написать скрипт, который будет, получив параметр N, запускать последовательно друг за другом N вычислений для разных значений входных параметров. Запуск вычисления для следующего параметра должен происходить сразу после завершения предыдущего вычисления.
- c. С помощью другого скрипта для каждого N в диапазоне от 1 до 20 запускать 10 раз запускающий скрипт из пункта 2 через утилиту time и фиксировать время, затраченное на полное выполнение запускающего скрипта. На выходе должно получиться 20 серий по 10 значений.
- d. В каждой серии посчитать среднее арифметическое значений и построить график зависимости этих усредненных оценок от N.

2. Второй эксперимент.

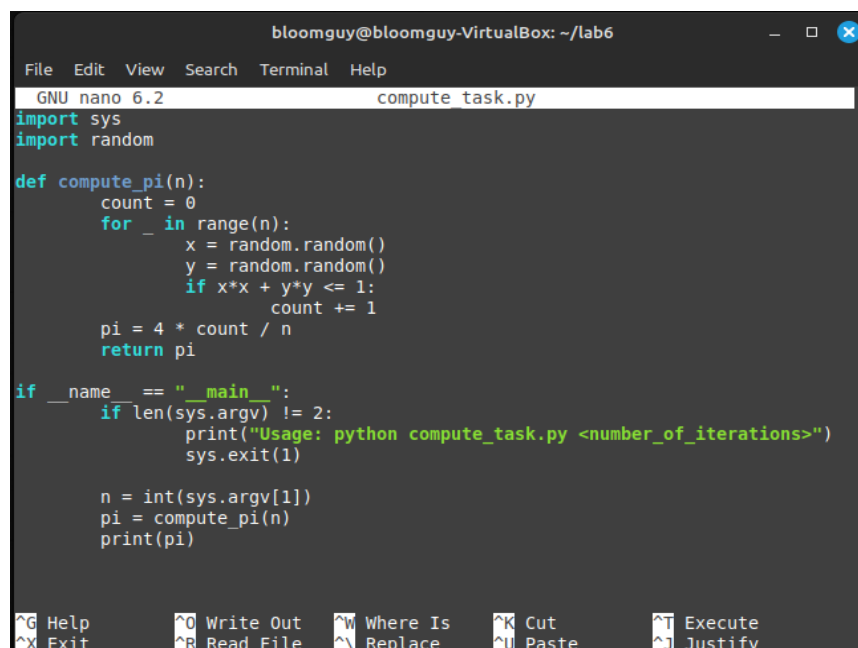
- a. Написать скрипт, который, получив параметр N, параллельно (не дожидаясь завершения предыдущего) запустит N вычислений для разных входных параметров.
- b. Повторить пункты “с” и “d” из первого эксперимента с использованием написанного в предыдущем пункте запускающего скрипта.

3. Третий и четвертый эксперимент – повторить первый и второй эксперимент, выделив в настройках виртуальной машины 2 процессора.

В результате выполнения этой группы экспериментов должны получиться 4 графика.

Первый и Второй эксперимент

compute_task.py (пример вычислительно сложного алгоритма):



```
bloomguy@bloomguy-VirtualBox: ~/lab6
File Edit View Search Terminal Help
GNU nano 6.2 compute_task.py
import sys
import random

def compute_pi(n):
    count = 0
    for _ in range(n):
        x = random.random()
        y = random.random()
        if x*x + y*y <= 1:
            count += 1
    pi = 4 * count / n
    return pi

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python compute_task.py <number_of_iterations>")
        sys.exit(1)

    n = int(sys.argv[1])
    pi = compute_pi(n)
    print(pi)

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

sequential_run.sh (Скрипт для последовательного выполнения вычислений):

```
bloomguy@bloomguy-VirtualBox: ~/lab6 — □ ×
File Edit View Search Terminal Help
GNU nano 6.2 sequential_run.sh
#!/bin/bash

if [ -z "$1" ]; then
    echo "Usage: $0 <number_of_tasks>"
    exit 1
fi

N=$1

for ((i=1; i<=N; i++))
do
    python3 compute_task.py $i
    if [ $? -ne 0 ]; then
        echo "Task $i failed" >&2
        exit 2
    fi
done

^G Help      ^O Write O   ^W Where I   ^K Cut
^X Exit      ^R Read Fi  ^\ Replace  ^U Paste
```

parallel_run.sh (Скрипт для параллельного выполнения вычислений):

```
bloomguy@bloomguy-VirtualBox: ~/lab6 — □ ×
File Edit View Search Terminal Help
GNU nano 6.2 parallel_run.sh
#!/bin/bash

if [ -z "$1" ]; then
    echo "Usage: $0 <number_of_tasks>"
    exit 1
fi

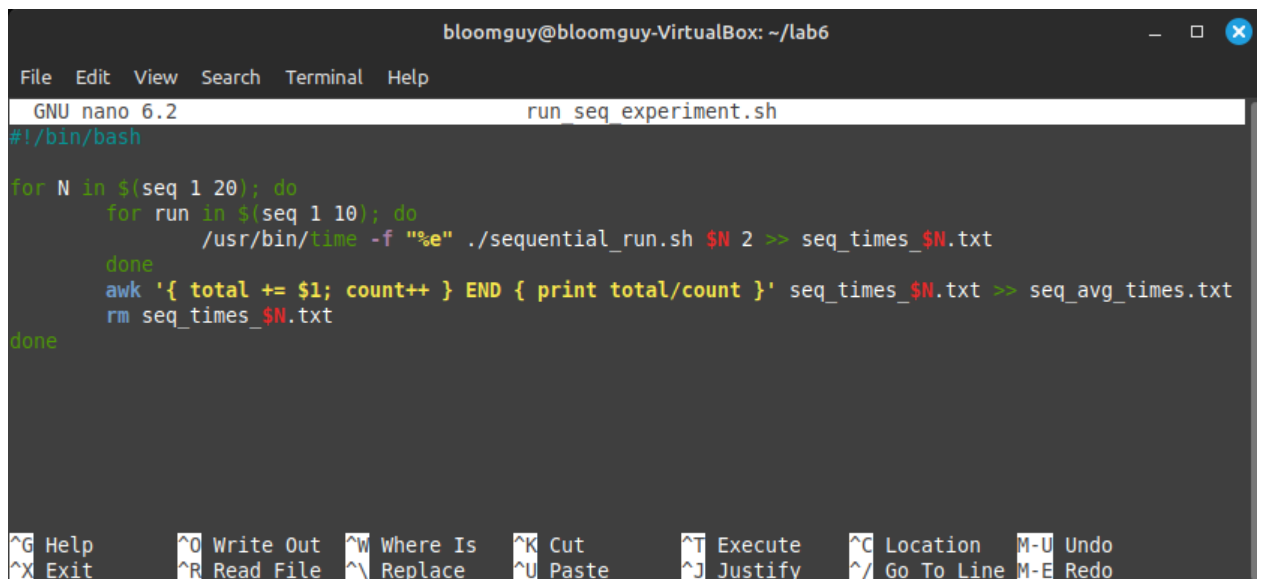
N=$1

for ((i=1; i<=N; i++))
do
    python3 compute_task.py $i &
done

wait
exit 0

[ Read 16 lines ]
^G Help      ^O Write O   ^W Where I   ^K Cut
^X Exit      ^R Read Fi  ^\ Replace  ^U Paste
```

run_seq_experiment.sh (Скрипт для проведения экспериментов с последовательным выполнением):



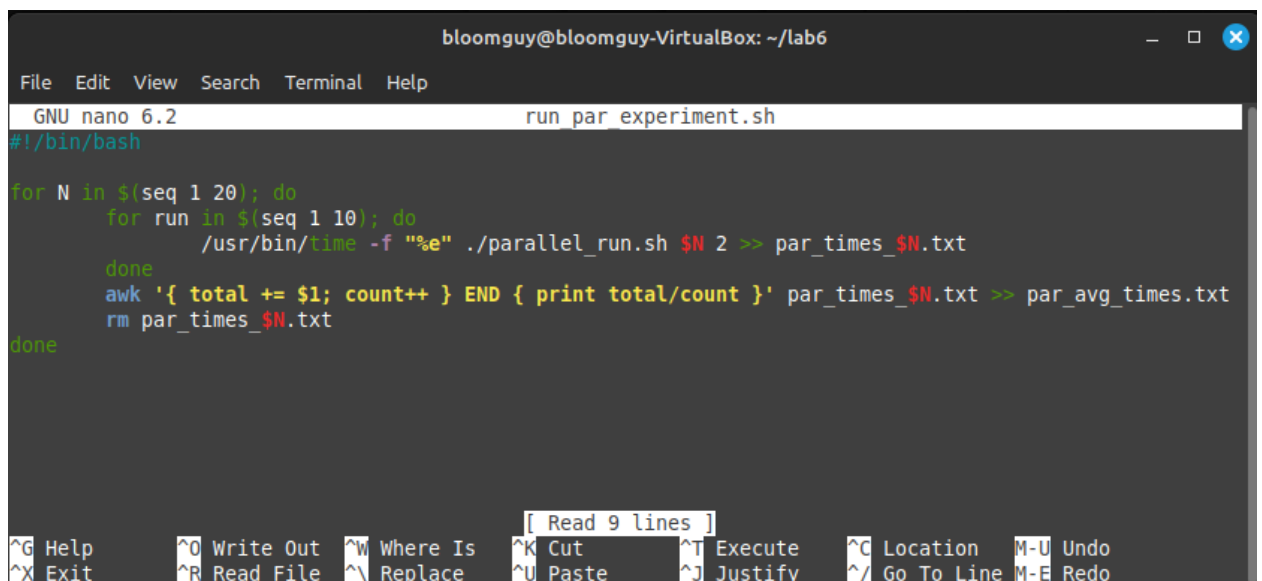
The screenshot shows a terminal window titled "bloomguy@bloomguy-VirtualBox: ~/lab6". The terminal is running the GNU nano 6.2 editor, editing the file "run_seq_experiment.sh". The script content is as follows:

```
#!/bin/bash

for N in $(seq 1 20); do
    for run in $(seq 1 10); do
        /usr/bin/time -f "%e" ./sequential_run.sh $N 2 >> seq_times_$N.txt
    done
    awk '{ total += $1; count++ } END { print total/count }' seq_times_$N.txt >> seq_avg_times.txt
    rm seq_times_$N.txt
done
```

The bottom of the window shows the nano editor's command palette with various shortcuts like ^G Help, ^O Write Out, ^W Where Is, etc.

run_par_experiment.sh (Скрипт для проведения экспериментов с параллельным выполнением):



The screenshot shows a terminal window titled "bloomguy@bloomguy-VirtualBox: ~/lab6". The terminal is running the GNU nano 6.2 editor, editing the file "run_par_experiment.sh". The script content is as follows:

```
#!/bin/bash

for N in $(seq 1 20); do
    for run in $(seq 1 10); do
        /usr/bin/time -f "%e" ./parallel_run.sh $N 2 >> par_times_$N.txt
    done
    awk '{ total += $1; count++ } END { print total/count }' par_times_$N.txt >> par_avg_times.txt
    rm par_times_$N.txt
done
```

The bottom of the window shows the nano editor's command palette. A tooltip "[Read 9 lines]" is visible over the "Paste" option.

plot_result.py (Скрипт для построения графиков зависимости среднего времени выполнения от количества задач):

```
bloomguy@bloomguy-VirtualBox: ~/lab6
File Edit View Search Terminal Help
GNU nano 6.2 plot_result.py
import matplotlib.pyplot as plt

def read_times(filename):
    with open(filename) as f:
        return [float(line.strip()) for line in f]

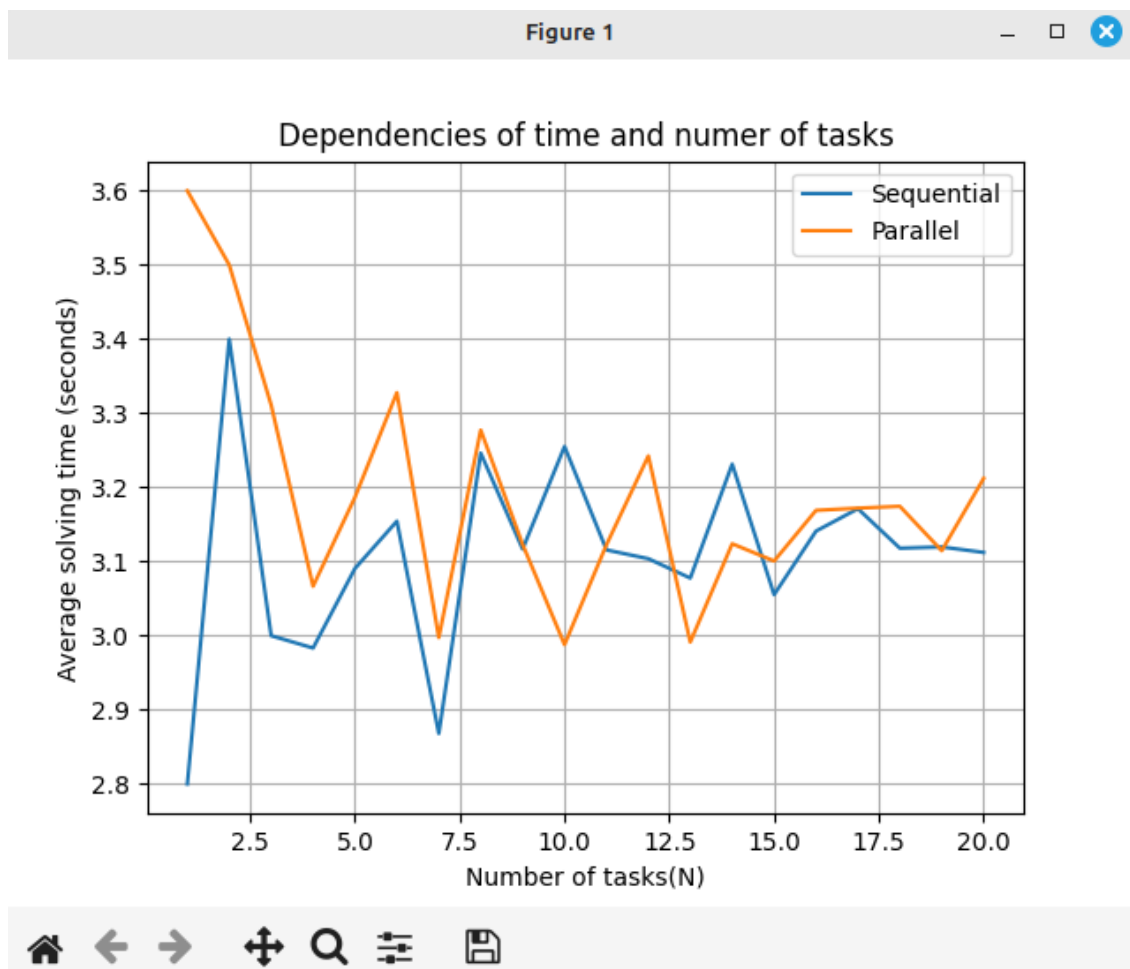
seq_times = read_times('seq_avg_times.txt')
par_times = read_times('par_avg_times.txt')

N_values = list(range(1, 21))

plt.plot(N_values, seq_times, label='Sequential')
plt.plot(N_values, par_times, label='Parallel')
plt.xlabel('Number of tasks(N)')
plt.ylabel('Average solving time (seconds)')
plt.title('Dependencies of time and number of tasks')
plt.legend()
plt.grid(True)
plt.savefig('result.png')
plt.show()

^G Help      ^O Write Out  ^W Where Is   ^K Cut
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```

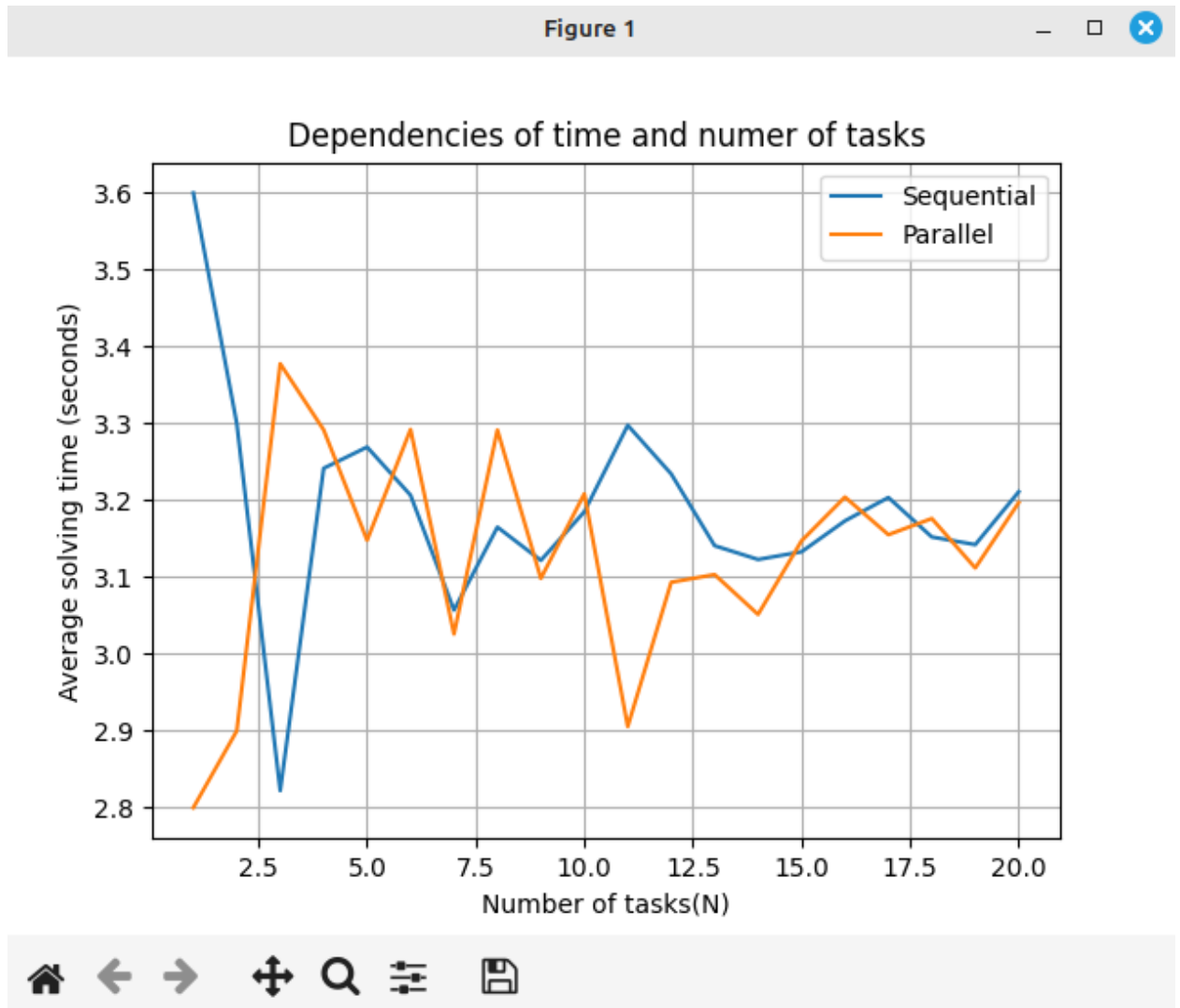
Графики, построенные из полученных ранее данных:



Третий и Четвёртый эксперимент

Для этих экспериментов нам потребуется проделать всё то же самое, только теперь в конфигурации виртуальной машины нужно выделить 2 ядра.

Графики:



Часть 2. Эксперимент с параллельным и последовательным выполнением задач с большими объемами считываемых и сохраняемых данных.

Для второй группы экспериментов необходимо реализовать алгоритм, который будет обрабатывать большие массивы данных, но с простыми вычислениями. Например, создать 20 файлов, размером по 10 МБайт (можно меньше, нужно смотреть по реальной производительности тестовой системы), представляющим последовательность целых чисел. Задачей алгоритма будет в одном файле умножить все значения на два и дописать значения в конце файла. Алгоритм должен считывать очередное значение из файла, умножать его на 2 и сохранять результат, дописывая в конце файла получившееся значение, а затем переходить к следующему значению и т.д. Важно реализовать именно такое взаимодействие с файлом:

чтение отдельного значения – изменение – сохранение измененного значения в файл. Нельзя формировать сразу все значения в памяти и потом записывать разом в файл – это будет уже другой эксперимент. Размер исходного файла нужно подобрать так, чтобы обработка одного файла занимала 2-3 секунды.

Далее нужно повторить эксперименты по плану из первой группы. Важно, чтобы каждый запущенный алгоритм работал со своим отдельным файлом.

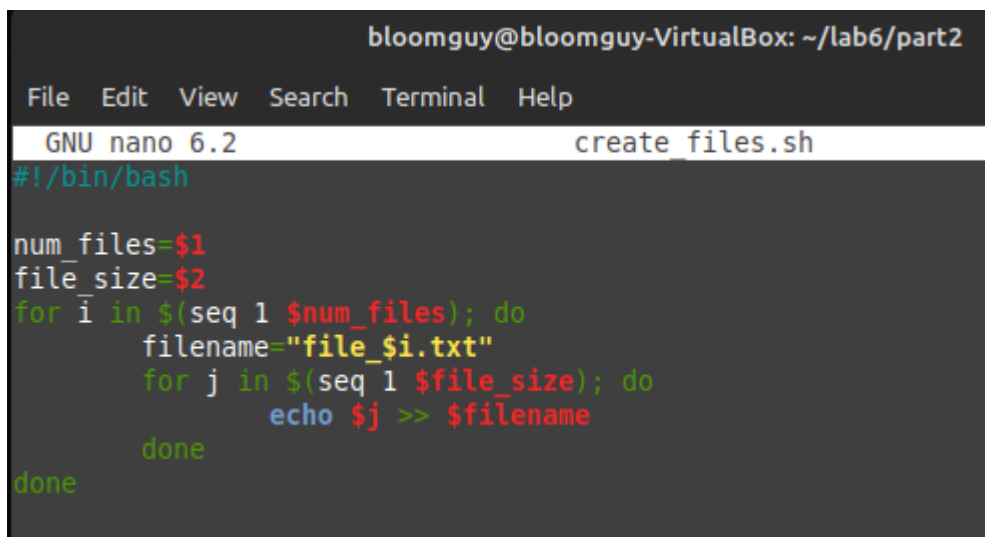
Результатом выполнения этой группы экспериментов будет еще 4 графика.

Можно и приветствуется, если Вы дополните эксперименты так, чтобы кроме оценки времени выполнения наборов заданий, оценивались и другие параметры ОС: память, потребляемые отдельными процессами ресурсы и т.д. Выбор таких параметров и методов их оценки на усмотрение экспериментаторов.

Шаг 1: Создание файлов с данными

Создает 20 файлов, которые примерно соответствуют 10 МБ.

Файл: **create_files.sh**:



```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 create_files.sh
#!/bin/bash

num_files=$1
file_size=$2
for i in $(seq 1 $num_files); do
    filename="file_${i}.txt"
    for j in $(seq 1 $file_size); do
        echo $j >> $filename
    done
done
```

Шаг 2: Скрипты для последовательного и параллельного выполнения с реализацией алгоритма для обработки файлов

Файл: **sequential_run.sh**:

```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 sequential_run.sh
#!/bin/bash

process_file() {
    local filename=$1
    local tempfile=$(mktemp)

    while read -r line; do
        echo $((line * 2)) >> "$tempfile"
    done < "$filename"
    cat "$tempfile" >> "$filename"
    rm "$tempfile"
}

N=$1
for i in $(seq 1 $N); do
    process_file "file_${i}.txt"
done
```

Файл: parallel_run.sh:

```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 parallel_run.sh
#!/bin/bash

process_file() {
    local filename=$1
    local tempfile=$(mktemp)

    while read -r line; do
        echo $((line * 2)) >> "$tempfile"
    done < "$filename"
    cat "$tempfile" >> "$filename"
    rm "$tempfile"
}

N=$1
for i in $(seq 1 $N); do
    process_file "file_${i}.txt" &
done
wait
```

Шаг 3: Скрипты для проведения экспериментов

Файл: run_seq_experiment.sh:


```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 run_seq_experiment.sh
#!/bin/bash

mkdir -p results

measure_time() {
    local mode=$1
    local N=$2
    local output_file=$3

    for run in {1..10}; do
        /usr/bin/time -f "%e" -o "$output_file" -a bash $mode $N
    done
}

for N in {1..20}; do
    rm -f results/seq_$N.txt

    measure_time "./sequential_run.sh" $N "results/seq_$N.txt"
done
```

Файл: run_par_experiment.sh:

```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 run_par_experiment.sh
#!/bin/bash

mkdir -p results

measure_time() {
    local mode=$1
    local N=$2
    local output_file=$3

    for run in {1..10}; do
        /usr/bin/time -f "%e" -o "$output_file" -a bash $mode $N
    done
}

for N in {1..20}; do
    rm -f results/par_$N.txt

    measure_time "./parallel_run.sh" $N "results/par_$N.txt"
done
```

Шаг 4: Построение графиков

Файл: plots.py:

```
bloomguy@bloomguy-VirtualBox: ~/lab6/part2
File Edit View Search Terminal Help
GNU nano 6.2 plots.py
import matplotlib.pyplot as plt
import numpy as np

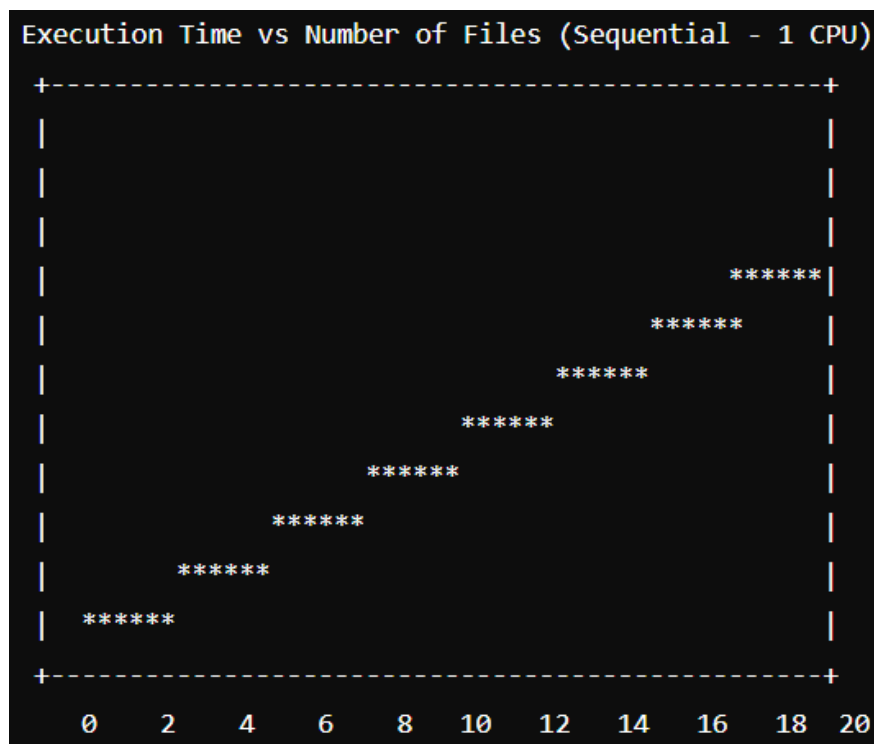
def read_times(filename):
    with open(filename) as f:
        times = [float(line.strip()) for line in f]
    return times

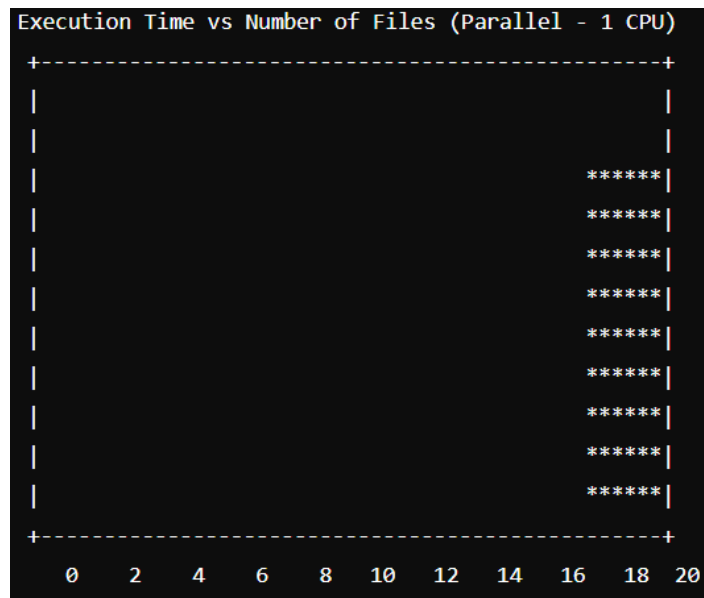
Ns = range(1, 4)
seq_means = [np.mean(read_times(f"results/seq_{N}.txt")) for N in Ns]
par_means = [np.mean(read_times(f"results/par_{N}.txt")) for N in Ns]

plt.figure(figsize=(10,6))
plt.plot(Ns, seq_means, label='Sequential', marker='o')
plt.plot(Ns, par_means, label='Parallel', marker='o')
plt.xlabel('Number of Files(N)')
plt.ylabel('Average Execution Time(s)')
plt.title('Execution Time vs Number of Files')
plt.legend()
plt.grid(True)
plt.show()
```

Повторение экспериментов с одним ядром и двумя ядрами:

С одним ядром:





С двумя ядрами:

