КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп’ютерних наук та кібернетики

Звіт

До лабораторної роботи

з дисципліни «Хмарні обчислення»

на тему:

Pancake sort з використанням технології PARCS

Виконав студент 4-го курсу
Факультету комп’ютерних наук та кібернетики
Кафедри ТТП
Расахацький Максим Володимирович

Київ – 2022

1. Постановка задачі

   Створити реалізацію розподіленого алгоритму сортування Pancake sort за допомогою технології PARCS (я використав PARCS Python), проаналізувати швидкість виконання від кількості воркерів.

2. Алгоритм виконання

   Скрипт реалізації має виконувати наступне:

   1. Зчитати масив випадкових чисел з файлу
   2. Розподілити його між воркерами
   3. Виконати сортування
   4. Об’єднати результат
   5. Вивести результат в файл

3. Скрипт

Функція розподілу:

```
 def solve(self):
        processed_arr = self.read_input()
        step = len(processed_arr) / len(self.workers)

        mapped = []
        for i in range(0, len(self.workers)):
            mapped.append(self.workers[i].mymap(processed_arr[i*step:i*step+step]
))

        reduced = self.myreduce(mapped)
        self.write_output(reduced)
```

Функція сортування:

```
@staticmethod
    @expose
    def mymap(array):
        n = len(array)
        curr_size = n
        while curr_size > 1:
            mi = findMax(array, curr_size)
            if mi != curr_size-1:
                flip(array, mi)
                flip(array, curr_size-1)
            curr_size -= 1
        return array


def flip(arr, i):
    start = 0
    while start < i:
        temp = arr[start]
        arr[start] = arr[i]
        arr[i] = temp
        start += 1
        i -= 1

def findMax(arr, n):
    mi = 0
    for i in range(0,n):
        if arr[i] > arr[mi]:
            mi = i
    return mi
```

Функція об’єднання

```
    @staticmethod
    @expose
    def myreduce(mapped):
        arr_num = len(mapped)
        res = mapped[0].value
        for i in range(1, arr_num):
            res = list(merge(res, list(mapped[i].value)))
        return res
```

Функції зчитування/запису:

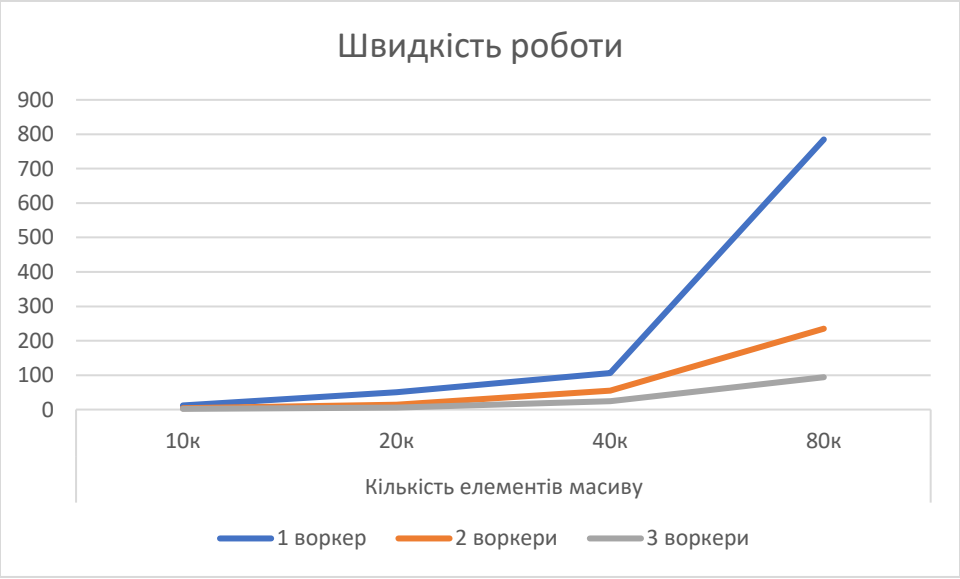
```
def read_input(self):
    f = open(self.input_file_name, 'r')
    array = []
    arr_line = f.readline().split(' ')
    for element in arr_line:
        if element != '':
            array.append(int(element))
    f.close()
    return array

def write_output(self, output):
    f = open(self.output_file_name, 'a')
    f.write(str(output))
    f.write('\n')
    f.close()
```

4. Результати виконання

Швидкість роботи, с

Кількість воркерів	**Кількість елементів масиву**			
	10к	**20к**	**40к**	**80к**
1	12	50	106	785
2	4	14	55	235
3	2	6	24	94



Проаналізувавши швидкість виконання можна побачити що для 80 тисяч елементів, використання 3 воркерів замість одного прискорює роботу більше ніж в 8 раз.

Висновки

Під час виконання лабораторної роботи було досліджено систему PARCS та використано її на практиці на платформі Google Cloud.

Алгоритм сортування млинцями може бути виконаний паралельно, що дозволяє його виконати більше ніж в 8 разів швидше. Різниця помітна на великій кількості даних.

Збільшення кількості воркерів зменшує швидкість виконання алгоритму, причому на розглянутому проміжку залежність не є лінійною.

Посилання на Github репозиторій

https://github.com/Rasakhatskiy/Labs_S7_PARCS