Київський національний університет

імені Тараса Шевченка



**Звіт**

**до лабораторної роботи**

*з курсу* "*Хмарні обчислення*"

*На тему:*

“Знаходження суми квадратів усіх чисел до заданого числа з використанням системи ПАРКС-Python”

**Студента четвертого курсу**

**Групи ТТП-41**

**Факультету комп’ютерних наук**

**та кібернетики**

**Грица Василя Васильовича**

***Київ***

*2022*

* **Постановка задачі**

З використанням системи ПАРКС-Python обчислити суму ряду який складається з квадратів чисел < за вхідне і проаналізувати прискорення, яке забезпечує використання кількох worker-ів.

* **Алгоритм**

Нехай ми маємо n > 0 воркерів. Спочатку ми розбиваємо вхідне число на n проміжків, які розподіляємо між воркерами. Метод *mymap* виконується на кожному воркері для свого проміжку. В ньому всі квадрати чисел в заданому проміжку сумуються. Далі метод *myreduce* об’єднує результати роботи всіх воркерів в один, просумувавши значення в усіх комірках, та повертає загальний результат.

* **Код програми**

Спочатку зчитуємо файл і кількість воркерів

def \_\_init\_\_(self, workers=None, input\_file\_name=None, output\_file\_name=None):

        self.input\_file\_name = input\_file\_name

        self.output\_file\_name = output\_file\_name

        self.workers = workers

На вхід дається ціле число і потрібно пошукати квадрати усіх натуральних чисел до того числа. Далі розподіл між воркерами

    def solve(self):

        n = self.read\_input()

        step = n / len(self.workers)

        mapped = []

        for i in xrange(0, len(self.workers)):

            mapped.append(self.workers[i].mymap(i \* step, i \* step + step))

        reduced = self.myreduce(mapped)

        self.write\_output(reduced)

На кожному воркері шукаємо суму квадратів чисел

  @staticmethod

    @expose

    def mymap(a, b):

        res = 0

        for i in xrange(a, b):

            res += pow(i, 2)

        return res

Далі зрощуємо усі суми які вийшли у воркерах і сумуємо

    @staticmethod

    @expose

    def myreduce(mapped):

        output = 0

        for x in mapped:

            output += x.value

        return output

Зчитування і запис у файл з результатом

    def read\_input(self):

        f = open(self.input\_file\_name, 'r')

        line = f.readline()

        f.close()

        return int(line)

    def write\_output(self, output):

        f = open(self.output\_file\_name, 'w')

        f.write(str(output))

        f.write('\n')

        f.close()

* **Отримані результати**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вхідне число  К-сть воркерів | 108 | 5\*108 | 109 | 3\*109 |
| 1 | 19с | 1хв 32с | 3хв 3с | 18хв 7с |
| 2 | 10с | 51с | 1хв 45с | 9хв 1с |
| 3 | 7с | 34с | 1хв 8с | 5хв 2с |

* **Висновок**

В ході виконання лабораторної роботи було вивчено та на практиці закріплено використання технології PARCS Python. Для хмарних обчислень було використано Google Cloud Platform.

Дослідивши результати виконання програм, можна зробити висновок , що зі збільшенням розміру вхідних даних в усіх трьох випадках зростає час виконання, а використання більшої кількості воркерів відповідно дає більше прискорення.

* **Посилання на GitHub**