

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №5
із дисципліни «Розподілені і хмарні обчислення»

Виконав:
студент групи КМ-01
Скорденко Д. О.

Керівник:
доцент кафедри ПМА
Ліскін В. О.

ЗМІСТ

Вступ	3
1 Основна частина	4
2 Опис програми [Тестовий приклад]	5
Висновки	7
Додаток Код лістинги	8

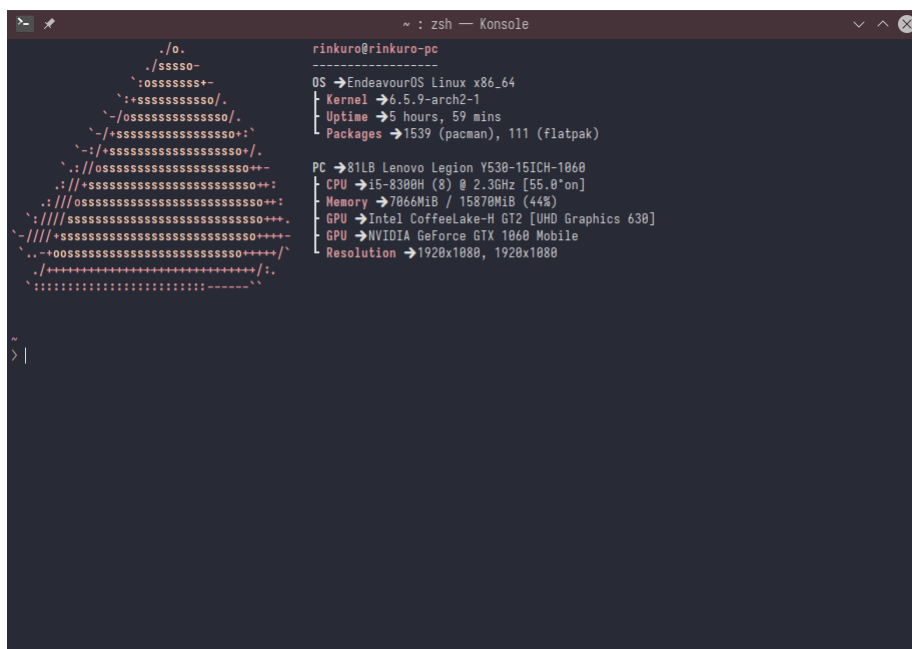
ВСТУП

Мета: розпаралелити метод обчислення константи π .

1 ОСНОВНА ЧАСТИНА

Опис програми: Для реалізації паралелізму буде використовуватись 'Rayon'.

2 ОПИС ПРОГРАМИ [ТЕСТОВИЙ ПРИКЛАД]

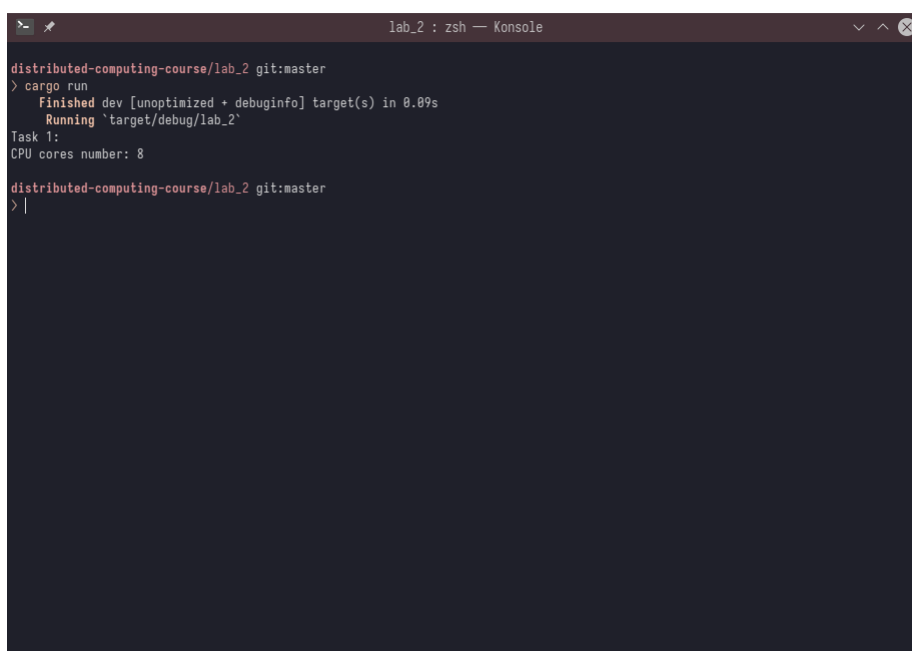


The screenshot shows a terminal window titled "zsh — Konsole" with the prompt "rinkuro@rinkuro-pc". On the left, there is a large ASCII art logo consisting of various symbols like dots, slashes, and asterisks arranged in a complex pattern. On the right, system information is displayed in a structured format:

```
rinkuro@rinkuro-pc
-----
OS → EndeavourOS Linux x86_64
Kernel → 6.5.9-arch2-1
Uptime → 5 hours, 59 mins
Packages → 1539 (pacman), 111 (flatpak)

PC → 81LB Lenovo Legion Y530-15ICH-1060
CPU → i5-8300H (8) @ 2.3GHz [55.0°C]
Memory → 7066MiB / 15870MiB (44%)
GPU → Intel CoffeeLake-H GT2 [UHD Graphics 630]
GPU → NVIDIA GeForce GTX 1060 Mobile
Resolution → 1920x1080, 1920x1080
```

Рисунок 2.1 – Характеристики системы



The screenshot shows a terminal window titled "lab_2 : zsh — Konsole" with the prompt "distributed-computing-course/lab_2 git:master". The user has entered the command "cargo run", and the output shows the following information:

```
distributed-computing-course/lab_2 git:master
> cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.09s
    Running `target/debug/lab_2`
Task 1:
CPU cores number: 8
distributed-computing-course/lab_2 git:master
> |
```

Рисунок 2.2 – К-сть ядер процессора

Таблиця 2.1 – Порівняння швидкодії

Метод	Результат
Редукція	3.1415928535894384
Монте-Карло	3.1415868

ВИСНОВКИ

Було розпаралелено два методи обчислення числа π .

Метод Монте-Карло виявився менш точним ніж метод редукції.

Додаток
Код лістинги

*Примітка: У код лістингах при копіюванні втрачається форматування (не копіюються пробіли). Файли прикріплено до цього pdf (вкладка "прикріплені файли").

Listing 1: constcalc.rs

```
use std::sync::{Arc, Mutex};
use rand::Rng;
use rayon::prelude::*;

pub fn picalc(n: i32) → f64 {
    let w = 1.0 / n as f64;
    let f = |x: f64| 4.0/(1.0+x*x);
    let sum = Arc::new(Mutex::new(0.0));

    let _sum = sum.clone();
    (0..n).into_par_iter().for_each(|i| {
        let x = w * (i as f64 - 0.5);
        *_sum.lock().unwrap() += f(x);
    });

    let pi = w * (*sum.lock().unwrap());
    pi
```



```
}
```

```
pub fn mc_picalc(n: i32) → f64 {
    let counter = Arc::new(Mutex::new(0 as i64));

    (0..n).into_par_iter().for_each(|_| {
        let x: f64 = rand::thread_rng().gen_range(-1.0..1.0);
        let y: f64 = rand::thread_rng().gen_range(-1.0..1.0);

        if x.powi(2) + y.powi(2) < 1.0 {
            let mut counter = counter.lock().unwrap();
            *counter += 1;
        }
    });

    let counter = counter.lock().unwrap();
    let pi = 4.0 * (*counter as f64) / n as f64;
    pi
}
```

Listing 2: lib.rs

```
pub mod constcalc;
```

Listing 3: main.rs

```
use lab_5::constcalc::{picalc, mc_picalc};
```

```
fn task_1() {  
    println!("Task 1");  
    let pi = picalc(1e+7 as i32);  
    println!("Picalc: {}", pi);  
}
```

```
fn task_2() {  
    println!("Task 2");  
    let pi = mc_picalc(1e+7 as i32);  
    println!("Monte-Carlo Picalc: {}", pi);  
}
```

```
fn main() {  
    task_1();  
    task_2();  
}
```