

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

Звіт  
із лабораторної роботи №2  
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

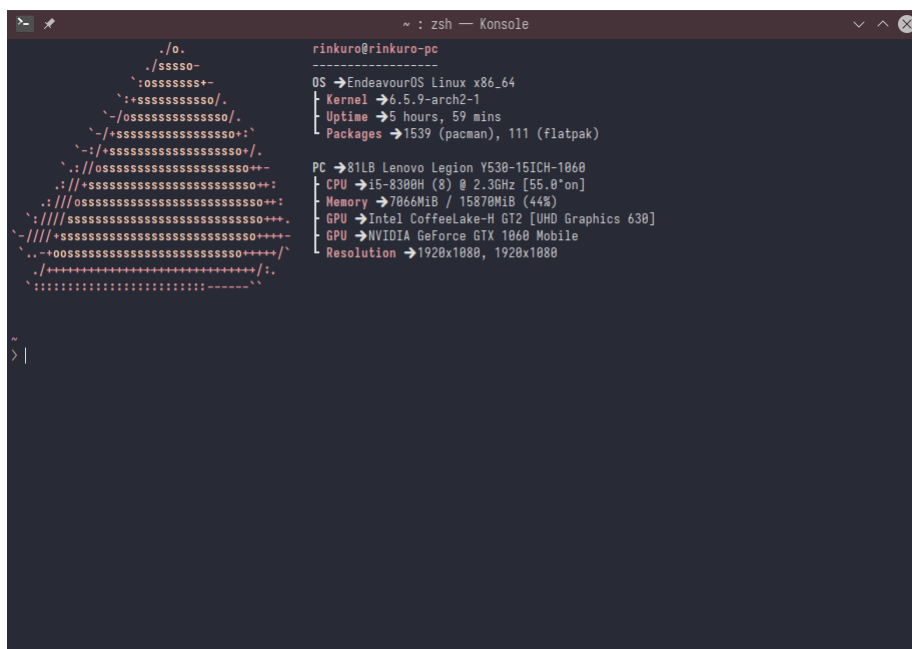
## ВСТУП

**Мета:** порівняти однопоточну та багатопоточну версії матричного множення та додавання елементів матриці.

## 1 ОСНОВНА ЧАСТИНА

**Опис програми:** Для реалізації паралелізму буде використовуватись 'Rayon'. Для порівняння швидкості обчислень буде використовуватись 'Criterion'.

## 2 ОПИС ПРОГРАМИ [ТЕСТОВИЙ ПРИКЛАД]

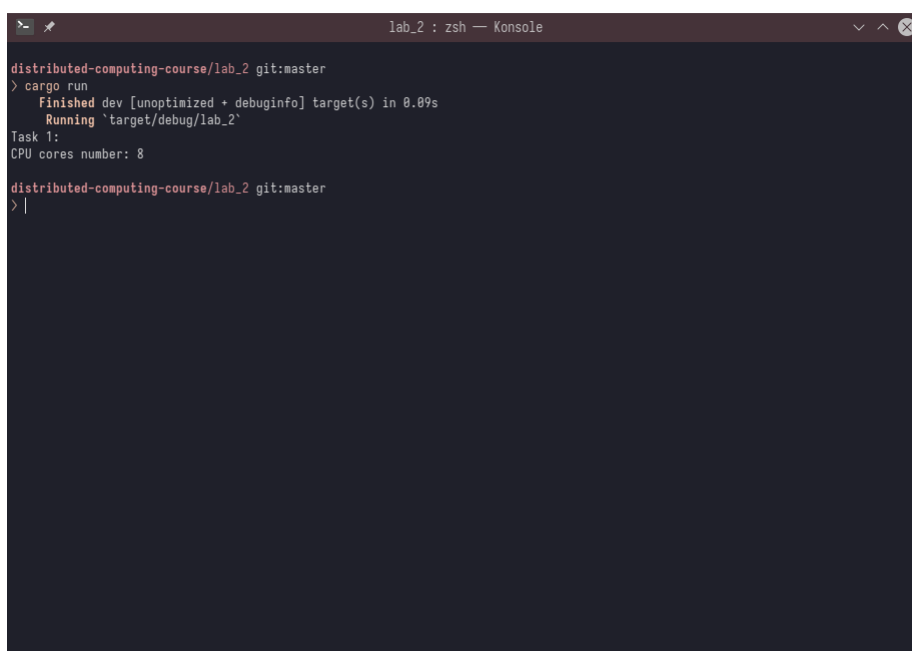


The screenshot shows a terminal window titled "zsh — Konsole" with the user "rinkuro@rinkuro-pc". On the left, there is a large ASCII art logo consisting of various symbols like dots, slashes, and plus signs arranged in a complex pattern. On the right, system information is displayed in a structured 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 user "distributed-computing-course/lab\_2 git:master". The terminal output shows the execution of a command and the resulting number of CPU cores:

```
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

Матричне множення 150x150 (Single thread)	[12.063 ms 12.267 ms 12.480 ms]
Матирчне множення 150x150 (Multi thread)	[3.3324 ms 3.3971 ms 3.4711 ms]
Додавання елементів матриці 1000x1000 (Single thread)	[3.6564 ms 3.6751 ms 3.6963 ms]
Додавання елементів матриці 1000x1000 (Multi thread)	[965.96 $\mu$ s 990.37 $\mu$ s 1.0199 ms]

## ВИСНОВКИ

Алгоритми із паралельним обчисленням (Multi thread) в середньому швидші на 50% – 75%.

## Додаток

## Код лістинги

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

## Listing 1: matop.rs

```
use rayon::prelude::*;
use nalgebra::{DMatrix};

pub fn matmul_paralel(m1: &DMatrix<f64>, m2: &DMatrix<f64>) → DMatrix<f64> {
    let m1shape = m1.shape();
    let m2shape = m2.shape();
    let nrows = m1shape.0;
    let ncols = m2shape.1;

    let iter: Vec<f64> = (0..m2shape.1).into_par_iter().flat_map(move |rj|
        (0..m1shape.0).into_par_iter().map(move |li| {
            (0..m2shape.0)
                .zip(0..m1shape.1)
                .map(move |(ri, lj)| {
                    m1.index((li, lj)) * m2.index((ri, rj))
                })
                .sum()
        })
    ).collect();

    DMatrix::from_vec(nrows, ncols, iter)
}
```



```
    })
  })
  .collect();
```

```
DMatrix::<f64>::from_iterator(nrows, ncols, iter)
}
```

```
pub fn matmul(m1: &DMatrix<f64>, m2: &DMatrix<f64>) -> DMatrix<f64> {
    let m1shape = m1.shape();
    let m2shape = m2.shape();
    let nrows = m1shape.0;
    let ncols = m2shape.1;

    let iter: Vec<f64> = (0..m2shape.1).flat_map(move |rj| {
        (0..m1shape.0).map(move |li| {
            (0..m2shape.0)
                .zip(0..m1shape.1)
                .map(move |(ri, lj)| {
                    m1.index((li, lj)) * m2.index((ri, rj))
                })
                .sum()
        })
    })

    .collect();
```

```

    DMatrix::<f64>::from_iterator(nrows, ncols, iter)
}

```

```

pub fn matelsum_paralel(m: &DMatrix<f64>) → f64 {
    let shape = m.shape();

    (0..shape.0).into_par_iter().map(|i| {
        m.row(i).sum()
    }).reduce(|| 0.0, |a,b| a + b)
}

```

```

pub fn matelsum(m: &DMatrix<f64>) → f64 {
    let shape = m.shape();

    (0..shape.0).into_iter().map(|i| {
        m.row(i).sum()
    }).reduce(|a,b| a + b).unwrap()
}

```

Listing 2: lib.rs

```
pub mod matop;
```

Listing 3: main.rs

```
use rayon::{current_num_threads};
```

```
fn task_1() {  
    println!("Task 1:");  
    let num: usize = current_num_threads();  
    println!("CPU cores number: {}", num);  
}
```

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