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: | Керівник: |
| студент групи КМ-01 | *Доцент кафедри ПМА*  *Ліскін В. О.* |
| *Романецький М.С.* |  |

Київ — 2023

Мета роботи: Порівняти однопоточну та багатопоточну версії матричного множення та додавання елементів матриці

Опис програми: Бібліотека ‘Rayon’ буде використовуватись для паралелізму. Для порівняння часу роботи буде використовувати ‘Criterion’

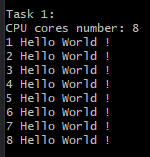


Характеристики системи

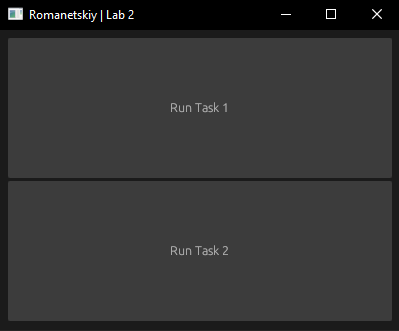
Кількість ядер процесора = 8

|  |  |  |
| --- | --- | --- |
|  |  | Середній час |
| Множення матриць | Однопоточна реалізація | 11.295 ms |
| Багатопоточна реалізація | 3.1764 ms |
| Додавання матриць | Однопоточна реалізація | 4.6789 ms |
| Багатопоточна реалізація | 1.3938 ms |

Отже багатопоточна реалізація в даному випадку працює швидше, іноді в декілька разів



Завдання 1



Інтерфейс програми

Код програми:

*// cd /d/KPI/Distributed\_computing/Labs/lab\_2*

use eframe::{epi, egui::{*self*, CtxRef}};

use rayon::{current\_num\_threads};

use std::process::Command;

use egui::vec2;

struct MyApp {

*//  дані та стан програми*

}

impl Default for MyApp {

    fn default() -> *Self* {

*Self* {

*// Ініціалізація стану*

        }

    }

}

impl epi::App for MyApp {

    fn name(&*self*) -> &str {

        "Romanetskiy | Lab 2"

    }

    fn update(&mut *self*, ctx: &CtxRef, \_frame: &mut epi::Frame) {

        egui::CentralPanel::default().show(ctx, |ui| {

            let button\_size = vec2(ui.available\_width(), 140.0);

            if ui.add\_sized(button\_size, egui::Button::new("Run Task 1")).clicked() {

                task\_1()

            }

            if ui.add\_sized(button\_size, egui::Button::new("Run Task 2")).clicked() {

                task\_2()

            }

        });

    }

}

fn main() {

    let app = MyApp::default();

    let mut native\_options = eframe::NativeOptions::default();

    native\_options.initial\_window\_size = Some(egui::vec2(400.0, 300.0));

    eframe::run\_native(Box::new(app), native\_options);

}

fn task\_1() {

    println!("\nTask 1:");

    let num\_cores: usize = current\_num\_threads();

    println!("CPU cores number: {num\_cores}");

    for i in 0..num\_cores {

        let n = i + 1;

        println!("{n} Hello World !");

    }

}

fn task\_2() {

    println!("\nTask 2:");

    println!("Запускаємо cargo bench");

    match Command::new("cargo").args(&["bench"]).status() {

        Ok(status) => if status.success() {

            println!("cargo bench виконано успішно");

        } else {

            eprintln!("cargo bench завершилося з помилкою");

        },

        Err(e) => eprintln!("Помилка при запуску cargo bench: {}", e),

    }

}