Індивідуальне завдання
Запуск контейнерів Docker у Kuberenetes

План

1. Теоретичні відомості
2. Завдання

1. Rust – це сучасна компільована мова програмування зі статичною типізацією, яка підтримує процедурну, функціональну та частково об’єктно-орієнтовану парадигми програмування. Початково мова Rust створювалася для системного програмування, яка може в перспективі замінити C/C++, але на сьогодні вона придатна для розв’язання широкого кола задач.

Rust було вперше офіційно представлено у 2010 році на Mozilla Summit 2010. Починаючи з 2015 року, кожні шість тижнів виходить нова версія, яка підтримує зворотну сумісність.

Останні сім років Rust регулярно займає перше місце в рейтингу найулюбленіших мов програмування, який складається Stack Overflow Developer Survey.

Свою назву мова Rust отримала на честь родини грибів Іржа, які є злісними шкідниками рослин.

Головними перевагами Rust у порівнянні з іншими сучасними мовами програмування є:

1) орієнтація на безпеку:
   – безпечна робота з пам’яттю (відсутність нульових покажчиків);
   – автоматичне керування пам’яттю завдяки концепції володіння, яке на відміну від традиційного збирання сміття (garbage collection) не витрачає системні ресурси;
2) орієнтація на паралельність і ефективність коду:
   – легкість в створенні нових потоків;
   – висока швидкість роботи;
   – прямий і простий інтерфейс для коду на мові C;
   – відсутність гонитви за даними у багатопотокових застосунках;
3) орієнтація на практичне застосування:
   – підтримка мультипарадигмальності;
   – підтримка функцій вищого порядку;
   – мультиплатформовість (працює у середовищі Windows, Linux, macOS тощо);
   – підтримка рядків у форматі UTF-8;
   – велике різноманіття вбудованих типів;
   – розвинена бібліотека;
   – підтримка тестів;
   – підтримка засобів низькорівневого програмування.

Завантажити Rust можна з офіційної сторінки за адресою: https://www.rust-lang.org/tools/install.

Особливістю Rust є використання спеціального пакетного менеджера **cargo**, який дозволяє виконувати управління проектами, компіляцію та запуск програм. Так, наприклад, створити новий проєкт програми «hello_world» за допомогою cargo можна скориставшись наступною командою: **cargo new hello_world**.

При створенні нового проєкту cargo автоматично генерує всі необхідні файли і каталоги:

– каталог src, у якому зберігається початковий текст програми – main.rs;

– конфігураційний файл проєкту – cargo.toml (файл маніфесту);

– допоміжні файли та каталоги системи керування версіями (git).

cargo.toml містить як всю необхідну для компіляції інформацію, так і додаткові відомості про програму (назву та версію проєкту, e-mail розробника, вживані зовнішні бібліотеки і т. д.). Конфігураційний файл описується із використанням спеціального формату TOML (Tom’s Obvious Minimal Language).

Наприклад:

```
[package]
name = "fem"
version = "0.1.0"
edition = "2018"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
ndarray = "*"
bitflags = "*"
chrono = "*"
rayon = "*"
json = "*"
russell_lab = "*"
russell_sparse = "*"
```

В розділі [dependencies] наводиться список необхідних для збирання проекту зовнішніх бібліотек і їх версій (символ «*» означає, що слід використовувати останню версію).

Компіляція проєкту у самому простому випадку здійснюється командою **cargo build**. При цьому cargo перевіряє, чи змінився проєкт після попередньої компіляції, і при необхідності завантажує зовнішні бібліотеки.

За замовченням cargo збирає налагоджувальну (debug) версію проекту. Для збирання остаточної версії слід скористатися наступною командою: **cargo build --release**. Для запуску програми можна скористатися командою **cargo run**, або безпосередньо її запустити з теки **target**, в яку компілятор записує всі проміжні файли та програму або бібліотеку, збирання яких і є кінцевою метою.

Для очищення проєкту від тимчасових файлів використовується команда **cargo clean**.

2. Завантажити проект на мові Rust, що реалізує певні розрахунки, зі сховища GitLab за посиланням: https://gitlab.com/gserega71/fem.git.

Створити образ Docker з підтримкою мови програмування Rust, в якому реалізується збирання проекту (команда **cargo build --release --bin example1**). При запуску контейнера виконувати відповідну програму, що знаходиться у теці ./target.

Слід враховувати, що для збирання цього проекту в Linux слід встановити бібліотеки liblapacke-dev, libmumps-seq-dev, libopenblas-dev і libsuitesparse-dev (наприклад, командою **sudo apt-get -y install liblapacke-dev libmumps-seq-dev libopenblas-dev libsuitesparse-dev**).

Запустити контейнер та зафіксувати час його виконання. Зібрати контейнер у налагоджувальній версії (без параметра --release) і порівняти час його виконання з попереднім варіантом.

Порівняти результати запуску контейнера без- та з використанням Kubernetes.

Самостійно розібратися з тим, як у програмі задається кількість обчислювальних потоків, що використовуються для розрахунків. Виконати розрахунок згідно з варіантом:

варіант 1 – 2 потоки;

варіант 2 – 4 потоки;

варіант 3 – 6 потоків;

варіант 4 – 8 потоків;

варіант 5 – 10 потоків.

Побудувати графік залежності часу виконання програми від кількості вживаних потоків (1 – 12). Пояснити отримані залежності.