Trabajo Práctico 2: Servidor HTTP Concurrente en Rust

Programación Concurrente - Primer cuatrimestre 2025

Profesores: Emilio Lopez Gabeiras y Rodrigo Pazos

18 de marzo de 2025

1. Introducción

Este trabajo práctico actualiza el TP1 para implementar un servidor HTTP concurrente en Rust. Se evoluciona el servidor anterior para gestionar eficientemente múltiples *requests* simultáneas, usando *threads* de la biblioteca estándar de Rust. El objetivo es analizar el comportamiento del servidor bajo carga concurrente.

2. Objetivos

- Actualizar el servidor HTTP para procesar *requests* de forma concurrente.
- Implementar el cálculo de la Serie de Liebniz hasta el término i para *requests* GET `/pi/:i` utilizando *threads*.
- Demostrar el manejo eficiente de múltiples *requests* concurrentes.
- Utilizar exclusivamente librerías estándar de Rust (`std`).

3. Pruebas

Para evaluar el servidor concurrente, utilizar `ab` (Apache Benchmark) para generar carga.

Prueba de concurrencia con `ab`:

```
$ ab -n 500 -c 50 http://localhost:3030/pi/1000000
```

(Envía 500 *requests*, con 50 concurrentes).

Probar con diferentes valores de i y carga concurrente (`-c` en `ab`) para verificar robustez y rendimiento.

4. Preguntas Abiertas

Una vez operativo, explorar:

1. Bajo carga concurrente intensa (aumentando `-n` y `-c` en `ab`), ¿qué efectos se observan en el comportamiento del servidor? ¿Se nota alguna diferencia en los tiempos de respuesta, la latencia o el comportamiento general? ¿A qué se debe?