Trabajo Práctico 5: Implementación de Queue Concurrente en Rust

Programación Concurrente - Primer cuatrimestre 2025

Profesores: Emilio Lopez Gabeiras y Rodrigo Pazos

6 de mayo de 2025

1. Introducción

Este trabajo práctico propone la implementación de una estructura de datos concurrente: una cola (*queue*) con soporte para múltiples productores y consumidores. Se debe desarrollar una versión bloqueante y otra no bloqueante utilizando únicamente las primitivas de concurrencia de la biblioteca estándar de Rust. El objetivo es analizar las diferencias de comportamiento y rendimiento entre ambas aproximaciones.

2. Objetivos

- Implementar una cola FIFO concurrente con múltiples productores y consumidores.
- Desarrollar dos variantes:
  - Una versión **bloqueante** basada en `Mutex` y `Condvar`.
  - Una versión **no bloqueante** utilizando técnicas de concurrencia sin bloqueo (lock-free), como el patrón de comparación e intercambio (`compare_and_swap`).
- Comparar el comportamiento de ambas versiones bajo diferentes niveles de concurrencia.

- Utilizar exclusivamente librerías estándar de Rust (`std`).

3. Pruebas

Para evaluar el rendimiento y la corrección de las colas, diseñar un programa de prueba donde múltiples hilos productores inserten datos en la cola, y múltiples hilos consumidores los extraigan.

Ejemplo de ejecución con 4 productores y 4 consumidores:

```
$ cargo run -- --producers 4 --consumers 4 --items 100000
```

Medir el tiempo total de ejecución y validar que todos los elementos insertados hayan sido consumidos correctamente, sin pérdidas ni duplicaciones. En principio, es válido únicamente probar que se haya consumido la misma cantidad de elementos que fueron insertados.

Comparar los resultados de la versión bloqueante y no bloqueante variando:

- Cantidad de elementos a insertar.
- Cantidad de hilos productores y consumidores.

4. Preguntas Abiertas

Una vez que ambas versiones estén funcionando correctamente, reflexionar:

1. ¿Qué diferencias observás en el rendimiento entre la versión bloqueante y la no bloqueante?
2. ¿Qué dificultades técnicas encontraste al implementar la versión no bloqueante?
3. ¿Bajo qué escenarios conviene usar cada una?
4. ¿Qué pasaría si se mezclan productores bloqueantes con consumidores no bloqueantes (o viceversa)?