

Control 2 — Lenguajes de Programación II

Tema: Ejecución Especulativa en Go

Profesor: Alonso Inostrosa Psijas

Fecha de entrega: 28/10/2025

Integrantes

- Thean Orlandi
 - Lucas Orellana
 - Angel Pino
-

Descripción del proyecto

Este proyecto implementa **ejecución especulativa** en el lenguaje **Go**, utilizando **goroutines** y **canales** para manejar la concurrencia y la sincronización.

El objetivo es comparar el rendimiento de una ejecución **secuencial** tradicional con una **especulativa**, donde ambas ramas de cómputo (A y B) se ejecutan en paralelo mientras se evalúa una condición costosa.

Concepto de ejecución especulativa

Se lanzan tareas en paralelo antes de saber cuál será necesaria.

Cuando se determina el resultado de la condición, la rama correcta se mantiene y la otra se **cancela** de manera controlada mediante `context.WithCancel()`.

Tecnologías y herramientas

- Lenguaje: **Go 1.20+**
 - Concurrencia: **Goroutines y Channels**
 - Cancelación: **context.Context**
 - Almacenamiento de métricas: **CSV**
 - Análisis: **Promedios y Speedup**
-

Estructura del proyecto

```
1 control2/
2   └─ main.go          # Código principal (modos spec, seq,
3     bench)
4   └─ go.mod           # Módulo Go
5   └─ control2.exe      # Ejecutable generado
6   └─ bench_metrics.csv # Resultados del benchmark
```

Instrucciones de uso

Compilación

```
1 go mod init control2
2 go build -o control2.exe main.go
3
```

Ejecución

Modo especulativo

```
1 ./control2.exe -mode spec -n 120 -umbral 800 -out
spec_metrics.csv -pow_diff 5 -primes_max 50000
2
```

Modo secuencial

```
1 ./control2.exe -mode seq -n 120 -umbral 800 -out
seq_metrics.csv -pow_diff 5 -primes_max 50000
2
```

Benchmark (30 repeticiones automáticas)

```
1 ./control2.exe -mode bench -runs 30 -n 120 -umbral 800 -out
bench_metrics.csv -pow_diff 5 -primes_max 50000
2
```

Parámetros del programa

Parámetro	Descripción	Ejemplo
-n	Dimensión de matrices NxN para la traza	120
-umbral	Umbral para decidir la rama ganadora	800
-pow_diff	Dificultad del Proof of Work	5
-primes_max	Límite superior para búsqueda de primos	50000
-mode	Modo de ejecución (spec, seq, bench)	bench

-runs	Número de repeticiones en modo benchmark	30
-out	Archivo CSV de salida con métricas	bench_metrics.csv

Análisis de rendimiento

Después de ejecutar el benchmark con **30 repeticiones por modo**, se obtuvieron los siguientes resultados promedio:

Modo	Tiempo promedio (ms)
Secuencial	1858.43
Especulativo	1414.93
Speedup	1.3134x

Interpretación

El resultado muestra un **Speedup > 1**, lo que significa que la ejecución especulativa fue **31% más rápida** que la secuencial.

Esto ocurre porque la función de decisión (CalcularTrazaDeProductoDeMatrices) tardó lo suficiente para permitir que las ramas especulativas avanzaran antes de conocerse la condición, aprovechando el solapamiento.

Conclusiones

1. **El patrón especulativo se implementó correctamente:**
 - Las ramas A y B se ejecutan concurrentemente.
 - Cuando se conoce la condición, se **cancela la rama perdedora** mediante `context.WithCancel()`.
 - Los resultados se comunican con canales (`chan`).
 2. **El rendimiento depende del costo relativo entre la condición y las ramas:**
 - Si la condición es costosa, la ejecución especulativa puede ofrecer **mejoras reales de rendimiento**.
 - Si la condición es corta, el overhead de concurrencia puede neutralizar el beneficio.
-