

Certamen II- Lenguajes de Programación II

Andrés González, andres.gonzalezvi@estudiantes.uv.cl

Lorena Uribe, lorena.uribe@estudiantes.uv.cl

Profesor: Alonso Inostrosa Psijas

El proyecto implementa una simulación multihilo basada en un esquema de sincronización optimista. Un hilo Scheduler genera eventos externos y los distribuye de forma cíclica entre varios hilos Workers. Cada Worker mantiene un reloj virtual local (LVT), procesa eventos externos, genera eventos internos, crea checkpoints y ejecuta rollbacks cada vez que detecta una violación de la causalidad, es decir, cuando lee un evento externo que fue enviado con un tiempo anterior a su LVT. Toda la actividad queda registrada en el archivo logs.csv, el cual se utiliza posteriormente para construir una visualización offline de la evolución del tiempo virtual en cada hilo.

A partir del archivo logs.csv se construyó un gráfico donde cada Worker aparece como una línea independiente, mostrando cómo evoluciona su LVT a medida que procesa eventos externos e internos.

Elementos del gráfico:

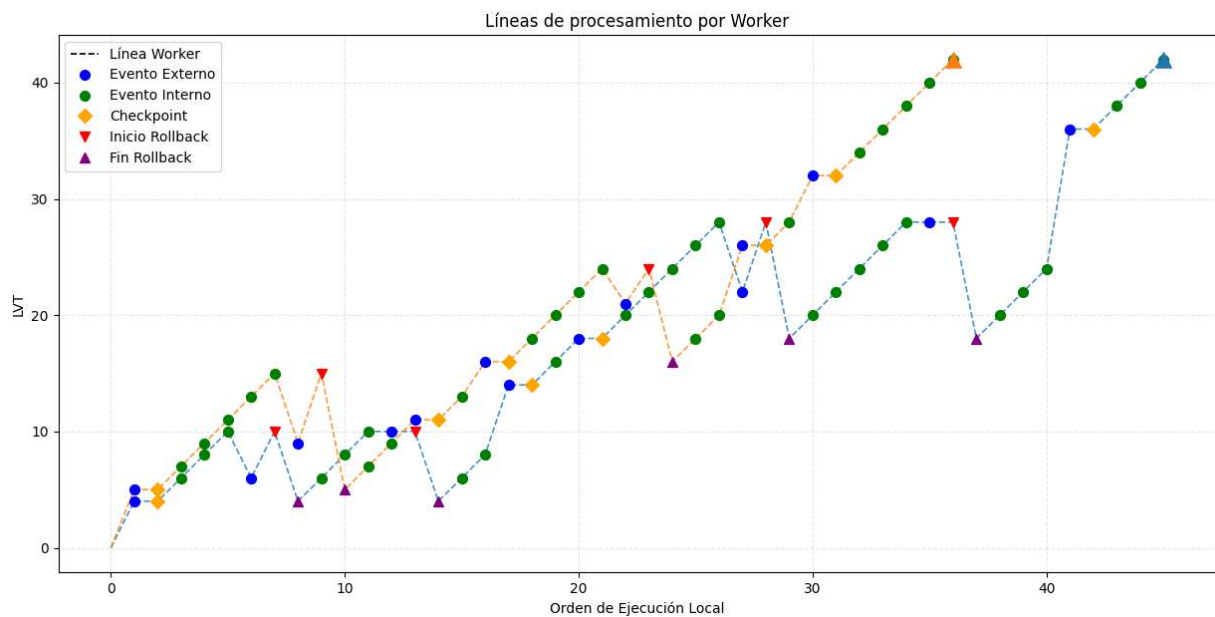
- Línea punteada: del color correspondiente a cada worker, muestra el camino que siguió durante su ejecución.
- Círculo azul: recepción y lectura de un evento enviado por Scheduler (externo).
- Círculo verde: evento generado dentro del Worker (interno).
- Rombo naranja: el Worker genera un checkpoint, guardando su estado interno.
- Triángulo rojo: comienzo de un rollback al encontrarse eventos con tiempos conflictivos.
- Triángulo morado: conclusión de un rollback, estado interno devuelto al del checkpoint inmediatamente posterior al tiempo del evento conflictivo.

Para ilustrar el comportamiento de los workers, se mostrará el gráfico resultante de la ejecución del programa. Esta ejecución se realizó con el siguiente comando:

```
go run main.go 2 15 4
```

Donde:

- <cantidad_workers>: número de Workers a generar.
- <cantidad_eventos>: cantidad de eventos externos a generar.
- <delay_eventos_max>: incremento máximo de tiempo para los timestamps de eventos.
- <seed>: opcional, para garantizar reproducibilidad de los resultados.



En este ejemplo, se generaron 2 workers y un total de 15 eventos externos, cada uno de ellos con una diferencia de entre 1 y 4 unidades de tiempo.

En el gráfico se aprecian ambas líneas de ejecución, cada una usando la simbología determinada para cada uno de los eventos realizados por cada worker.

Prueba de Escalabilidad

La simulación fue ejecutada con diferentes cantidades de Workers: 1, 2, 4, 8, y 16 Workers. A continuación, se muestran los tiempos promedio de ejecución y el speedup obtenido al comparar con el tiempo de ejecución de 1 Worker.

Tabla

Número de Workers	Tiempos (ms)	Tiempo Promedio (ms)	Speedup contra 1 Worker
1	22.8572, 28.6003, 28.3753, 28.4621, 25.7789, 28.0042, 28.3085	27.198	1.00
2	28.1807, 23.8678, 27.0063, 25.6374, 25.1352, 27.9303, 28.6978	266.365	10.211
4	27.7086, 23.3417, 28.2296, 25.5142, 27.6558, 24.9671, 25.4392	261.152	10.415
8	23.8873, 23.4676, 25.6935, 25.3963, 28.9964, 27.6086, 24.9588	242.869	11.199
16	22.3901, 22.0845, 21.2672, 24.1670, 23.4981, 24.7568, 25.0415	233.150	11.665

Cálculo del Speedup:

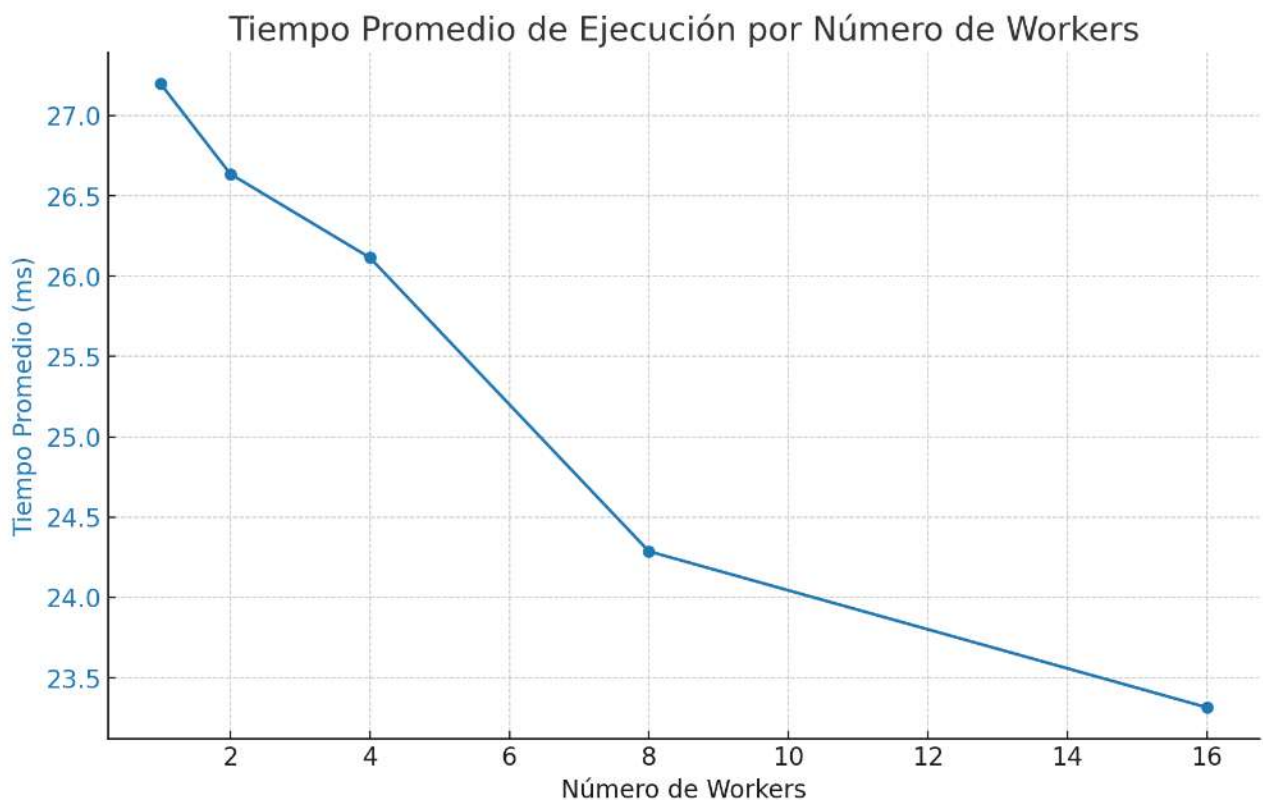
Speedup: es el cociente entre el tiempo de ejecución con 1 Worker y el tiempo de ejecución con N Workers.

La fórmula es: $\text{Speedup} = T_1 / T_n$

Análisis de los Resultados:

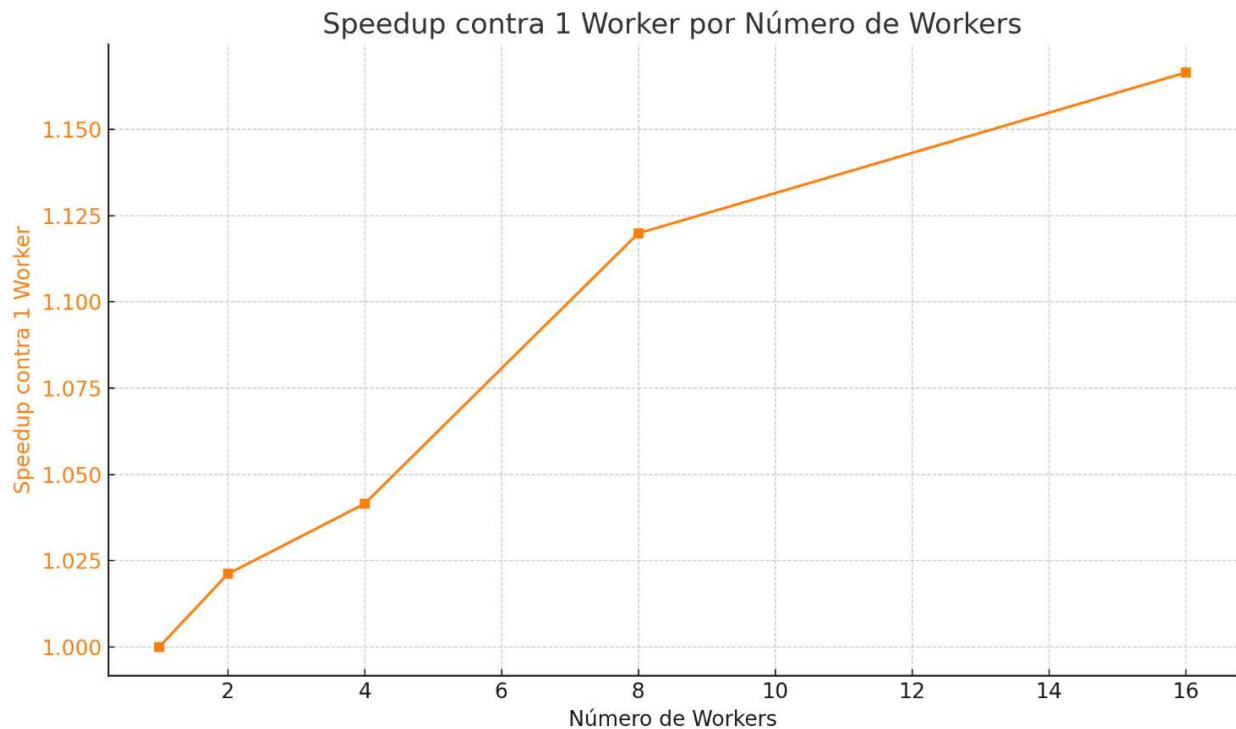
- 1 Worker: Caso base, sin interferencia entre hilos. Promedio de tiempo: 27.198 ms.
- 2 Workers: Aumento de Speedup a 10.211. A pesar de que la ejecución se distribuye mejor, se observa mayor sobrecarga debido a los rollbacks.
- 4 Workers: Mejor desempeño con un Speedup de 10.415. El balance de carga mejora, pero el costo de los checkpoints sigue presente.
- 8 Workers: La mejora es más leve, con Speedup de 11.199, donde el sistema ya muestra signos de saturación de paralelismo.
- 16 Workers: El Speedup alcanza 11.665. A partir de aquí, el aumento de paralelismo no genera mejoras significativas debido a la sobrecarga de los rollbacks.

Gráfico de Tiempo Promedio de Ejecución por Número de Workers:



Muestra cómo el tiempo promedio de ejecución disminuye a medida que aumentan los Workers.

Gráfico de Speedup contra 1 Worker:



Muestra cómo el Speedup aumenta con la adición de más Workers, pero con una mejora cada vez menor.

Conclusión

El sistema muestra mejoras en el rendimiento con el aumento de Workers, especialmente al pasar de 1 a 2 y 4 Workers. Sin embargo, a partir de 8 Workers y 16 Workers, el speedup se estabiliza, indicando que el sistema no sigue escalando de manera eficiente debido a la sobrecarga de checkpoints y rollbacks. El máximo speedup alcanzado fue 11.665, lo que sugiere que más Workers no siempre se traducen en un rendimiento significativamente mejor. La sobrecarga de mantener la causalidad limita las ganancias de paralelismo a partir de cierto punto.