Informe de Línea de Ensamblaje con IPC, Sincronización y Algoritmos de Scheduling

Descripción de la solución

El proyecto implementa una simulación de una línea de ensamblaje de productos utilizando el lenguaje de programación **Rust**. En esta línea, los productos pasan por varias estaciones (`Corte`, `Ensamblaje` y `Empaque`), donde cada estación tiene un tiempo de procesamiento definido.

La comunicación entre estaciones se realiza mediante **canales (`mpsc`)** y cada estación se ejecuta en un **hilo (`thread`)** separado, permitiendo procesamiento concurrente.

El sistema utiliza dos algoritmos de planificación para el manejo de los productos en cada estación:

- **First Come First Serve (FCFS)**: Los productos se procesan estrictamente en el orden en que llegan.
- **Round Robin (RR)**: Los productos se procesan en rebanadas de tiempo (`quantum`) definidas. Si no terminan en el quantum asignado, se vuelven a poner al final de la cola para continuar su procesamiento.

El usuario puede seleccionar el algoritmo de planificación al iniciar el programa y, en caso de elegir Round Robin, también puede configurar el valor del `quantum`.

Al final de la simulación, se calcula y muestra un resumen con:

- El tiempo de llegada de cada producto.
- El tiempo de entrada y salida de cada producto en cada estación.
- El tiempo de espera total de cada producto.
- El tiempo de *turnaround* (tiempo total en el sistema) de cada producto.
- El promedio de tiempos de espera y *turnaround* para todos los productos.
- El orden final en que los productos terminaron su procesamiento.

Justificación Técnica

- **Rust y concurrencia segura**: Se eligió Rust por su capacidad para manejar concurrencia de manera segura y eficiente. El uso de `Arc` y `Mutex` garantiza que el acceso compartido a datos entre hilos sea seguro, evitando condiciones de carrera.
- **Canales (`mpsc`)**: Se utilizaron canales para transmitir los productos entre las estaciones de forma asíncrona y segura, desacoplando así la lógica de procesamiento de cada estación.
- **Modelado limpio**: El diseño orientado a objetos mediante estructuras (`struct`) para `Station`, `Product`, y `ProcessingStep` facilita el mantenimiento y la extensión futura del proyecto.
- **Algoritmos de planificación**:
  - **FCFS**: Es simple de implementar y sirve como línea base (baseline) para evaluar el rendimiento del sistema.
  - **Round Robin**: Permite simular situaciones donde se necesita repartir el tiempo de procesamiento de manera más equitativa entre varios productos, siendo especialmente útil cuando hay productos que requieren tiempos largos de procesamiento.

- **Flexibilidad**: El sistema permite fácilmente cambiar o agregar nuevos algoritmos de planificación en el futuro, gracias a la separación de lógica según el tipo de algoritmo (`SchedulingAlgorithm`).

Comparación entre algoritmos

Criterio	FCFS	Round Robin
Orden de procesamiento	Estrictamente por orden de llegada.	Basado en rebanadas de tiempo (`quantum`).
Tiempo de espera	Puede ser alto para productos que llegan tarde.	Más equilibrado; productos largos no bloquean tanto.
Equidad	Baja: los primeros acaparan la estación.	Alta: todos reciben atención periódicamente.
Complejidad implementación	Más sencillo.	Más complejo (manejo de tiempo restante, reencolado).
Uso recomendado	Cargas de trabajo homogéneas y simples.	Cargas variadas, con productos de tiempos distintos.

Resultados esperados

- **FCFS** tiende a favorecer a los primeros productos y puede provocar largos tiempos de espera para los siguientes, especialmente si los primeros productos requieren mucho tiempo de procesamiento (efecto convoy).
- **Round Robin** mejora la equidad, evitando que productos pequeños esperen demasiado, pero puede incrementar el *overhead* (sobrecarga por cambios de contexto) si el `quantum` no es bien elegido.

Resultados experimentales con diferentes configuraciones

Comparación de Rendimiento

Se ejecutaron tres configuraciones para comparar el rendimiento, utilizando el mismo conjunto de productos:

Métrica	FCFS	RR (quantum=500ms)	RR (quantum=1000ms)
Tiempo espera promedio	4.41s	16.11s	9.59s
Turnaround promedio	7.73s	19.44s	12.91s
Tiempo último producto	17.15s	24.61s	18.88s

Hallazgos clave

- **Impacto del Quantum**: Round Robin con `quantum` de 1000ms redujo el tiempo de espera promedio en un **40%** y el *turnaround* promedio en un **33.5%** frente al `quantum` de 500ms.
- **Eficiencia vs. Equidad**: FCFS fue significativamente más eficiente en términos de promedios de espera y *turnaround* para esta carga de trabajo específica. Sin embargo, Round Robin (especialmente con 1000ms) mostró una mayor equidad en la distribución de los tiempos de espera individuales.

Comportamiento por Producto

FCFS

- **Productos tempranos** (ej. Producto 4): Experimentan espera mínima (0s) y *turnaround* rápido (3.3s).
- **Productos tardíos** (ej. Producto 8): Sufren espera acumulativa alta (9.8s) debido al efecto convoy.
- **Variabilidad de turnaround alta**: La diferencia entre el *turnaround* máximo y mínimo fue de 13.2s.

Round Robin (quantum=1000ms)

- **Distribución más uniforme**: Los tiempos de espera se agruparon más, oscilando entre 9.4s y 12.1s para la mayoría de los productos.
- **Turnaround máximo reducido**: El *turnaround* máximo fue de 15.4s (comparado con 24.6s usando `quantum`=500ms), indicando una mejor gestión de los productos más largos.
- **Variabilidad de turnaround reducida**: La diferencia entre el *turnaround* máximo y mínimo fue menor que en FCFS, demostrando mayor equidad.

Optimización del Quantum del Round Robin

Se determinó que un `quantum` de **1000ms** es el más adecuado para esta simulación en comparación con 500ms y 1500ms:

- **Regla del 80% (aproximación)**: Citando 'Operating System Concepts', un buen `quantum` debería permitir que ~80% de las ráfagas de CPU terminen en un solo `quantum`.
  - Con `quantum=1000ms`, el 66% de los pasos de procesamiento (considerando tiempos de 1s, 1.2s, 1.1s) completaron su trabajo en 1 `quantum`. Esto se acerca razonablemente al objetivo.
  - Con `quantum=500ms`, **ningún** paso de procesamiento terminaba en un solo `quantum`, generando un promedio de 7 cambios de contexto por producto, lo cual incrementa significativamente el *overhead*.
- **Reducción de Overhead**: Pasar de 500ms a 1000ms redujo los cambios de contexto promedio por producto de 7 a 4, lo que explica la drástica disminución en los tiempos promedio de espera (16.11s → 9.59s, **-40%**) y *turnaround* (19.44s → 12.91s, **-33.5%**).
- **Equilibrio práctico**: El `quantum` de 1000ms mantiene una equidad notable (tiempos de espera similares para la mayoría de productos) sin sacrificar excesivamente la eficiencia global, a diferencia del `quantum` de 500ms que era muy ineficiente.

Análisis del Quantum de 1500ms

Se realizó un análisis adicional con `quantum=1500ms`:

1. **Comportamiento casi idéntico a FCFS**:
   - FCFS: Espera prom = 4.41s | Turnaround prom = 7.73s
   - RR (quantum=1500ms): Espera prom = 5.61s | Turnaround prom = 8.92s
   - La diferencia mínima se debe principalmente al *overhead* inherente de Round Robin, no a una gestión diferente de los productos.
2. **Orden de procesamiento similar**: Ambos algoritmos terminaron los productos prácticamente en el mismo orden (ej. Producto 8 primero en salir de la última estación, Producto 3 último).

3. **¿Por qué no es útil?**:
    - **Pierde la esencia de Round Robin**: Con un `quantum` tan largo que supera el tiempo de procesamiento de la mayoría de las tareas, no se logra la rotación ni la equidad esperada. Los productos tardíos siguen esperando casi tanto como en FCFS.
    - **Complejidad innecesaria**: Implementar Round Robin es más complejo que FCFS. Si con un `quantum` grande solo se replica el comportamiento de FCFS (pero con ligero *overhead* adicional), no aporta valor.

Conclusiones

- **FCFS** es ideal para cargas de trabajo predecibles y homogéneas donde la eficiencia (menor *turnaround* promedio: 7.73s) es prioritaria sobre la equidad estricta.
- **Round Robin** con un `quantum` bien ajustado (en este caso, **1000ms**) ofrece un buen equilibrio entre equidad y rendimiento. Logró reducir la variabilidad del *turnaround* y los tiempos de espera máximos comparado con FCFS, aunque con un costo en la eficiencia promedio.
- **Optimización del Quantum es Clave**: La elección del `quantum` en Round Robin tiene un impacto drástico en el rendimiento (hasta un ~40% de diferencia en espera promedio entre 500ms y 1000ms). Debe calibrarse cuidadosamente según los tiempos de procesamiento esperados para cumplir los objetivos (eficiencia vs. equidad).
- **Quantum Demasiado Grande**: Un `quantum` excesivamente largo (como 1500ms en este caso) hace que Round Robin se comporte casi como FCFS, perdiendo sus ventajas de equidad y añadiendo complejidad innecesaria.
- La simulación valida que **no hay un "mejor" algoritmo universal**; la elección depende de las características de la carga de trabajo y los objetivos de rendimiento (minimizar promedio, minimizar máximo, maximizar equidad).