ALGORITMO MLQ (Multi-Level Queue Scheduling)

Informe Técnico sobre Planificación de Procesos

---

1. INTRODUCCIÓN

El algoritmo **Multi-Level Queue (MLQ)** es una técnica de planificación de procesos utilizada en sistemas operativos que organiza los procesos en múltiples colas con diferentes prioridades y algoritmos de scheduling. Este enfoque permite optimizar el rendimiento del sistema al aplicar estrategias específicas según las características de cada tipo de proceso.

---

2. DESCRIPCIÓN DEL ALGORITMO

2.1 Concepto Fundamental

El MLQ divide los procesos en diferentes colas basándose en sus características o prioridades. Cada cola tiene su propio algoritmo de planificación y una prioridad fija. Los procesos no pueden moverse entre colas una vez asignados, lo que diferencia este algoritmo del Multi-Level Feedback Queue (MLFQ).

2.2 Estructura de Colas Implementada

En la implementación analizada, el sistema cuenta con tres colas con prioridad descendente:

COLA 1 (Prioridad Alta)

- Algoritmo: Round Robin con Quantum = 1
- Propósito: Procesos interactivos que requieren respuesta rápida

COLA 2 (Prioridad Media)

- Algoritmo: Round Robin con Quantum = 3
- Propósito: Procesos de prioridad media con mayor quantum

COLA 3 (Prioridad Baja)

- Algoritmo: Shortest Job First (SJF)
- Propósito: Procesos batch de baja prioridad

2.3 Funcionamiento del Algoritmo

El scheduler MLQ opera bajo los siguientes principios:

1. **Asignación Estática:** Cada proceso se asigna a una cola específica al momento de su llegada, basándose en su atributo de cola predefinido.
2. **Prioridad de Ejecución:** El sistema siempre ejecuta procesos de la cola de mayor prioridad disponible. Solo cuando la Cola 1 está vacía se procesa la Cola 2, y solo cuando ambas están vacías se ejecuta la Cola 3.
3. **Algoritmos por Cola:**
    - **Cola 1 (RR q=1):** Para procesos interactivos que requieren respuesta rápida. El quantum pequeño garantiza tiempos de respuesta cortos.
    - **Cola 2 (RR q=3):** Para procesos de prioridad media con quantum más largo, reduciendo el overhead de cambio de contexto.
    - **Cola 3 (SJF):** Para procesos batch de baja prioridad, minimizando el tiempo promedio de espera mediante la ejecución del trabajo más corto primero.

---

3. MÉTRICAS DE RENDIMIENTO

El sistema calcula las siguientes métricas para evaluar el desempeño del algoritmo:

Métrica	Descripción	Fórmula
WT	Waiting Time (Tiempo de Espera)	TAT - BT
CT	Completion Time (Tiempo de Finalización)	Tiempo cuando el proceso termina
RT	Response Time (Tiempo de Respuesta)	Primera ejecución - AT
TAT	Turnaround Time (Tiempo de Retorno)	CT - AT

Donde: BT = Burst Time (tiempo de ráfaga), AT = Arrival Time (tiempo de llegada)

---

4. FLUJO DE EJECUCIÓN

Paso 1: Los procesos se leen del archivo de entrada y se ordenan por tiempo de llegada.

Paso 2: En cada unidad de tiempo, se agregan a sus respectivas colas los procesos que han llegado al sistema.

Paso 3: Se selecciona la cola de mayor prioridad que no esté vacía.

Paso 4: Se ejecuta el siguiente proceso según el algoritmo de la cola seleccionada.

Paso 5: Si el proceso completa su ejecución, se calculan sus métricas y se marca como completado. Si no termina (en colas RR), regresa al final de su cola.

Paso 6: Se repite el proceso hasta que todos los procesos hayan completado su ejecución.

---

5. VENTAJAS Y DESVENTAJAS

5.1 Ventajas

- Permite aplicar diferentes estrategias de scheduling según el tipo de proceso
- Reduce los tiempos de respuesta para procesos interactivos de alta prioridad
- Optimiza el throughput para procesos batch mediante SJF
- Estructura simple y predecible
- Bajo overhead computacional

5.2 Desventajas

- **Starvation:** Los procesos de baja prioridad pueden sufrir inanición si hay flujo constante de procesos de alta prioridad
- Falta de flexibilidad: los procesos no pueden cambiar de cola
- Requiere clasificación previa de los procesos
- Puede ser injusto para procesos de baja prioridad

---

6. IMPLEMENTACIÓN EN JAVA

La implementación consta de los siguientes componentes principales:

6.1 Clase Process

Representa un proceso individual con sus atributos:

- ID del proceso
- Tiempo de ráfaga (Burst Time)
- Tiempo de llegada (Arrival Time)
- Cola asignada (1, 2 o 3)
- Prioridad
- Métricas calculadas (WT, CT, RT, TAT)

6.2 Clase ProcessQueue

Gestiona cada cola individual:

- Mantiene la lista de procesos en la cola

- Implementa la lógica específica del algoritmo (RR o SJF)
- Ordena procesos por prioridad dentro de la cola
- Gestiona el quantum para Round Robin

6.3 Clase MLQScheduler

Coordina todo el sistema:

- Gestiona las tres colas
- Controla el tiempo del sistema
- Decide qué cola ejecutar según prioridad
- Calcula métricas promedio
- Genera salida de resultados

6.4 Clase FileHandler

Maneja entrada/salida:

- Lee procesos desde archivo de texto
- Escribe resultados con métricas calculadas
- Formato: etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT

---

7. CONCLUSIONES

El algoritmo MLQ implementado demuestra ser una solución eficiente para sistemas que requieren diferenciar el tratamiento de procesos según su naturaleza. La combinación de Round Robin para procesos interactivos y SJF para procesos batch optimiza tanto el tiempo de respuesta como el throughput del sistema.

Sin embargo, es importante considerar mecanismos adicionales como aging o promotion policies en implementaciones productivas para evitar la inanición de procesos de baja prioridad, especialmente en sistemas con alta carga de trabajo en las colas prioritarias.

El sistema implementado proporciona una base sólida para comprender los conceptos de planificación multinivel y puede ser extendido para incluir características adicionales como:

- Mecanismos anti-starvation
- Colas dinámicas (MLFQ)
- Ajuste automático de quantums
- Prioridades adaptativas