Algoritmo de Planificación Multi-Level Feedback Queue (MLFQ)

Joshua Emmanuel Mendez Ospina

Sistemas Operativos

Profesor Jefferson Amado Peña Torres

20 de septiembre de 2025

Índice

1. Introducción

El algoritmo **Multi-Level Feedback Queue (MLFQ)** es uno de los algoritmos de planificación de procesos más sofisticados y ampliamente utilizados en sistemas operativos modernos. Este algoritmo combina las ventajas de diferentes estrategias de planificación para optimizar tanto el tiempo de respuesta para procesos interactivos como la eficiencia general del sistema.

1.1. Propósito del Algoritmo

El MLFQ fue diseñado para resolver varios problemas fundamentales en la planificación de procesos:

- **Optimizar el tiempo de respuesta**: Proporcionar tiempos de respuesta rápidos para procesos interactivos y de corta duración.
- **Maximizar el rendimiento**: Asegurar que los procesos de larga duración no monopolicen el CPU.
- **Adaptabilidad**: Ajustarse dinámicamente al comportamiento de los procesos sin conocimiento previo de sus características.
- **Prevención de inanición**: Garantizar que todos los procesos eventualmente reciban tiempo de CPU.

1.2. Características Principales

El MLFQ se caracteriza por:

- **Múltiples colas de prioridad**: El sistema mantiene varias colas, cada una con un nivel de prioridad diferente.
- **Retroalimentación dinámica**: Los procesos pueden moverse entre colas basándose en su comportamiento de ejecución.
- **Algoritmos híbridos**: Cada cola puede implementar un algoritmo de planificación diferente (Round Robin, SJF, STCF).
- **Quantum variable**: Los diferentes niveles pueden tener diferentes tamaños de quantum de tiempo.

2. Explicación Detallada del Algoritmo

2.1. Estructura de Múltiples Colas

El MLFQ organiza los procesos en múltiples colas de prioridad, típicamente numeradas desde el nivel más alto (1) hasta el más bajo (4 en nuestro simulador). La estructura es la siguiente:

- **Nivel 1 (Prioridad más alta)**: Procesos interactivos y de respuesta rápida
- **Nivel 2**: Procesos de prioridad media-alta
- **Nivel 3**: Procesos de prioridad media-baja
- **Nivel 4 (Prioridad más baja)**: Procesos de larga duración y tareas de fondo

2.2. Políticas de Planificación por Cola

En nuestro simulador, se implementan tres esquemas diferentes de configuración:

Esquema	Nivel 1	Nivel 2	Nivel 3	Nivel 4
A	RR(1)	RR(3)	RR(4)	SJF
B	RR(2)	RR(3)	RR(4)	STCF
C	RR(3)	RR(5)	RR(6)	RR(20)

Cuadro 1: Configuraciones de algoritmos por nivel

Donde:

- **RR(n)**: Round Robin con quantum de n unidades de tiempo
- **SJF**: Shortest Job First (no apropiativo)
- **STCF**: Shortest Time to Completion First (apropiativo)

2.3. Manejo de Prioridades y Movimiento entre Colas

2.3.1. Reglas de Prioridad

1. **Regla de prioridad absoluta**: Los procesos en niveles superiores siempre tienen prioridad sobre los de niveles inferiores.
2. **Apropiación inmediata**: Si llega un proceso a un nivel superior mientras se ejecuta uno de nivel inferior, se produce una apropiación inmediata.
3. **Degradación por quantum**: Cuando un proceso agota su quantum completo en Round Robin, se degrada al siguiente nivel.

2.3.2. Algoritmo de Movimiento

El movimiento entre colas sigue estas reglas:

- **Llegada inicial**: Todos los procesos comienzan en el Nivel 1 (prioridad más alta).
- **Degradación**: Si un proceso consume todo su quantum en Round Robin, se mueve al siguiente nivel inferior.
- **Terminación**: Si un proceso termina antes de agotar su quantum, permanece en el mismo nivel para futuras ejecuciones.

2.4. Algoritmos de Planificación Implementados

2.4.1. Round Robin (RR)

El algoritmo Round Robin asigna un quantum fijo de tiempo a cada proceso. Las características son:

- Cada proceso recibe una porción equitativa de tiempo de CPU
- Si el proceso no termina en su quantum, se coloca al final de la cola
- Ideal para sistemas de tiempo compartido e interactivos
- El quantum se incrementa en niveles inferiores para reducir el overhead de cambio de contexto

2.4.2. Shortest Job First (SJF)

SJF es un algoritmo no apropiativo que:

- Selecciona el proceso con el menor tiempo de ejecución restante
- Una vez iniciado, el proceso se ejecuta hasta completarse
- Minimiza el tiempo de espera promedio
- Se utiliza típicamente en el nivel más bajo para procesos de larga duración

2.4.3. Shortest Time to Completion First (STCF)

STCF es la versión apropiativa de SJF:

- Siempre ejecuta el proceso con menor tiempo restante
- Puede apropiar el CPU cuando llega un proceso con menor tiempo restante
- Óptimo para minimizar el tiempo de retorno promedio
- Proporciona mejor respuesta que SJF para cargas de trabajo variables

2.5. Ventajas y Desventajas

2.5.1. Ventajas

- **Adaptabilidad**: Se ajusta automáticamente a diferentes tipos de procesos sin configuración previa.
- **Buen tiempo de respuesta**: Los procesos interactivos reciben prioridad alta y respuesta rápida.
- **Eficiencia**: Los procesos de larga duración se manejan eficientemente en niveles inferiores.

- **Flexibilidad**: Permite combinar diferentes algoritmos según las necesidades de cada nivel.
- **Prevención de inanición**: Los mecanismos de envejecimiento pueden implementarse para evitar la inanición.

2.5.2. Desventajas

- **Complejidad de implementación**: Requiere mantener múltiples colas y políticas de movimiento.
- **Overhead administrativo**: El mantenimiento de múltiples colas introduce overhead del sistema.
- **Configuración sensible**: El rendimiento depende significativamente de la configuración de quantums y políticas.
- **Posible inanición**: Sin mecanismos adicionales, los procesos pueden quedar atrapados en niveles bajos.

3. Ejemplos Prácticos

3.1. Ejemplo 1: Caso mlq001.txt

3.1.1. Datos de Entrada

Para el primer ejemplo, utilizaremos el archivo `mlq001.txt` que contiene los siguientes procesos:

Proceso	Burst Time	Arrival Time	Queue Inicial	Prioridad
A	6	0	1	5
B	9	0	1	4
C	10	0	2	3
D	15	0	2	3
E	8	0	3	2

Cuadro 2: Procesos del ejemplo mlq001.txt

3.1.2. Ejecución con Esquema A

El Esquema A utiliza la configuración: RR(1), RR(3), RR(4), SJF.
Análisis paso a paso:

1. **Tiempo 0**: Todos los procesos llegan simultáneamente
   - Procesos A y B se asignan al Nivel 1 (RR con quantum=1)
   - Procesos C y D se asignan al Nivel 2 (RR con quantum=3)
   - Proceso E se asigna al Nivel 3 (RR con quantum=4)

2. **Nivel 1 (RR quantum=1)**:
   - A ejecuta 1 unidad, se degrada al Nivel 2 (tiempo restante: 5)
   - B ejecuta 1 unidad, se degrada al Nivel 2 (tiempo restante: 8)
3. **Nivel 2 (RR quantum=3)**:
   - C ejecuta 3 unidades, se degrada al Nivel 3 (tiempo restante: 7)
   - D ejecuta 3 unidades, se degrada al Nivel 3 (tiempo restante: 12)
   - A ejecuta 3 unidades, se degrada al Nivel 3 (tiempo restante: 2)
   - B ejecuta 3 unidades, se degrada al Nivel 3 (tiempo restante: 5)
4. **Nivel 3 (RR quantum=4)**:
   - E ejecuta 4 unidades, se degrada al Nivel 4 (tiempo restante: 4)
   - C ejecuta 4 unidades, se degrada al Nivel 4 (tiempo restante: 3)
   - D ejecuta 4 unidades, se degrada al Nivel 4 (tiempo restante: 8)
   - A ejecuta 2 unidades, termina
   - B ejecuta 4 unidades, se degrada al Nivel 4 (tiempo restante: 1)
5. **Nivel 4 (SJF)**:
   - B termina (tiempo restante: 1)
   - C termina (tiempo restante: 3)
   - E termina (tiempo restante: 4)
   - D termina (tiempo restante: 8)

3.1.3. Resultados del Esquema A

Proceso	BT	AT	Q Final	Pr	WT	CT	RT	TAT
A	6	0	3	5	16	22	0	22
B	9	0	4	4	30	39	1	39
C	10	0	4	3	31	41	2	41
D	15	0	4	3	33	48	3	48
E	8	0	3	2	30	38	4	38

Cuadro 3: Resultados del Esquema A para mlq001.txt

Métricas del sistema:

- Tiempo de Espera Promedio (WT): 28.0
- Tiempo de Finalización Promedio (CT): 37.6
- Tiempo de Respuesta Promedio (RT): 2.0
- Tiempo de Retorno Promedio (TAT): 37.6

3.2. Ejemplo 2: Caso mlq002.txt

3.2.1. Datos de Entrada

El segundo ejemplo utiliza el archivo `mlq002.txt`:

Proceso	Burst Time	Arrival Time	Queue Inicial	Prioridad
p1	20	0	1	5
p2	10	2	1	4
p3	15	4	2	3
p4	5	6	3	2
p5	8	8	3	1

Cuadro 4: Procesos del ejemplo mlq002.txt

3.2.2. Características del Caso 2

Este ejemplo presenta características más complejas:

- **Llegadas escalonadas**: Los procesos no llegan simultáneamente
- **Distribución de colas**: Los procesos se distribuyen en diferentes niveles iniciales
- **Variedad de tamaños**: Diferentes duraciones de burst time (5 a 20 unidades)

3.2.3. Ejecución con Esquema A

Secuencia temporal detallada:

1. **Tiempo 0-2**: Solo p1 está presente
   - p1 ejecuta en Nivel 1 (RR quantum=1): 1 unidad
   - p1 se degrada al Nivel 2, ejecuta 1 unidad más
2. **Tiempo 2**: Llega p2
   - p2 entra al Nivel 1 y tiene prioridad sobre p1
   - p2 ejecuta 1 unidad, se degrada al Nivel 2
3. **Tiempo 4**: Llega p3
   - p3 entra directamente al Nivel 2
   - Continúa la ejecución en Nivel 2 con p1, p2, y p3
4. **Tiempo 6**: Llega p4 al Nivel 3
5. **Tiempo 8**: Llega p5 al Nivel 3

3.2.4. Resultados del Esquema A

Proceso	BT	AT	Q Final	Pr	WT	CT	RT	TAT
p1	20	0	4	5	38	58	0	58
p2	10	2	4	4	27	39	2	37
p3	15	4	4	3	27	46	4	42
p4	5	6	3	2	18	29	6	23
p5	8	8	3	1	21	37	8	29

Cuadro 5: Resultados del Esquema A para mlq002.txt

Métricas del sistema:

- Tiempo de Espera Promedio (WT): 26.2
- Tiempo de Finalización Promedio (CT): 41.8
- Tiempo de Respuesta Promedio (RT): 4.0
- Tiempo de Retorno Promedio (TAT): 37.8

3.3. Comparación de Esquemas

3.3.1. Análisis Comparativo mlq001.txt

Esquema	WT	CT	RT	TAT
A	28.0	37.6	2.0	37.6
B	27.8	37.4	4.0	37.4
C	27.2	36.8	6.0	36.8

Cuadro 6: Comparación de esquemas para mlq001.txt

3.3.2. Análisis Comparativo mlq002.txt

Esquema	WT	CT	RT	TAT
A	26.2	41.8	4.0	37.8
B	26.0	41.6	4.0	37.6
C	30.2	45.8	6.0	41.8

Cuadro 7: Comparación de esquemas para mlq002.txt

3.4. Observaciones Importantes

- **Esquema A**: Ofrece el mejor tiempo de respuesta debido al quantum pequeño en el nivel superior
- **Esquema B**: Proporciona un balance entre tiempo de respuesta y eficiencia usando STCF en el nivel inferior

- **Esquema C**: Muestra mayor tiempo de respuesta pero puede ser más eficiente para cargas de trabajo específicas

4. Diagramas de Gantt

4.1. Diagrama de Gantt para mlq001.txt (Esquema A)

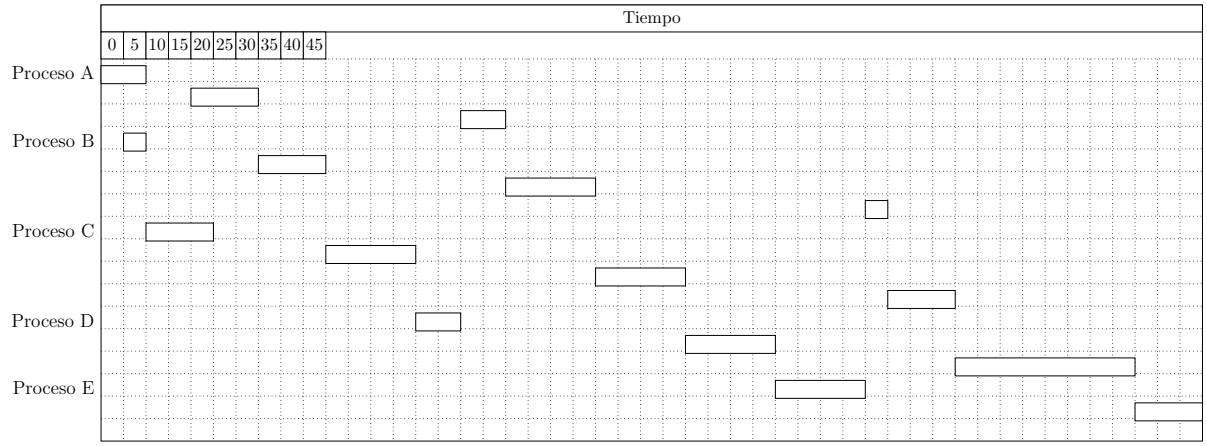

Figura 1: Diagrama de Gantt para mlq001.txt con Esquema A

5. Conclusiones

5.1. Importancia del MLFQ

El algoritmo Multi-Level Feedback Queue representa una solución elegante y práctica para los desafíos de planificación en sistemas operativos modernos. Su importancia radica en varios aspectos fundamentales:

- **Versatilidad**: Puede adaptarse a una amplia variedad de cargas de trabajo sin requerir conocimiento previo de las características de los procesos.
- **Equilibrio**: Logra un balance efectivo entre tiempo de respuesta para procesos interactivos y eficiencia para procesos de larga duración.
- **Implementación práctica**: Se utiliza en sistemas operativos reales como Linux, Windows y macOS con variaciones específicas.

5.2. Insights Clave de los Ejemplos

Del análisis de los casos de estudio se obtienen las siguientes conclusiones importantes:

5.2.1. Comportamiento de Degradación

- Los procesos que requieren más tiempo de CPU tienden a degradarse hacia niveles inferiores

- Esta degradación es beneficiosa porque permite que los procesos cortos e interactivos mantengan prioridad alta
- El sistema se autorregula sin intervención externa

5.2.2. Impacto de la Configuración

- El tamaño del quantum en cada nivel afecta significativamente el rendimiento
- Quantums pequeños en niveles altos mejoran la respuesta pero aumentan el overhead
- La elección del algoritmo en el nivel inferior (SJF vs STCF vs RR) impacta las métricas finales

5.2.3. Métricas de Rendimiento

Los resultados demuestran que:

- **Tiempo de Respuesta**: El Esquema A ofrece el mejor tiempo de respuesta (2.0-4.0) debido a su quantum agresivo
- **Tiempo de Espera**: Los tres esquemas muestran tiempos de espera similares, indicando estabilidad del algoritmo
- **Eficiencia General**: Las diferencias entre esquemas son menores, sugiriendo que MLFQ es robusto ante variaciones de configuración

5.3. Aplicaciones Prácticas

El conocimiento adquirido del análisis de MLFQ tiene aplicaciones directas en:

- **Administración de sistemas**: Comprender cómo ajustar parámetros de planificación en sistemas reales
- **Desarrollo de software**: Escribir aplicaciones que interactúen eficientemente con el planificador del OS
- **Análisis de rendimiento**: Diagnosticar problemas de rendimiento relacionados con la planificación de procesos
- **Diseño de sistemas**: Tomar decisiones informadas sobre configuraciones de sistemas críticos

5.4. Reflexiones Finales

El estudio del algoritmo MLFQ revela la elegancia de los sistemas que pueden autoorganizarse y adaptarse dinámicamente. Esta capacidad de aprendizaje y ajuste automático es fundamental en el diseño de sistemas operativos modernos, donde la variedad y complejidad de las cargas de trabajo requieren soluciones flexibles y robustas.

Los ejemplos analizados demuestran que, aunque la configuración específica puede afectar las métricas de rendimiento, el comportamiento fundamental del algoritmo permanece estable y predecible. Esta estabilidad, combinada con la capacidad de adaptación, hace del MLFQ una de las contribuciones más significativas en el campo de la planificación de procesos.

El simulador desarrollado proporciona una herramienta valiosa para comprender estos conceptos de manera práctica y experimental, permitiendo la exploración de diferentes configuraciones y escenarios de manera controlada y reproducible.