Assignment - 1

Name: Md. Minhazul Mowla

ID: 23201390

Section: 23

Course: CSE321

Submission Date: 13/11/25

CPU Allocation Fainness Issues in Multi-Level Feedback Queue (MLFQ) Scheduling

The MLFQ scheduling algorithm is mostly used in operating systems to ensure balance responsiveness and thruput. Even though MLFQ is effective in many cases, it has some fainness issues and specifically between I/O-bound and CPU-bound processes.

This report shows comprehensive analysis focusing on the following two key factors:

i. Unfair CPU Allocation due to I/O-bound processes

ii. Quantitative fainness Evalnation

i. Unfair CPU Allocation due to I/O-Bound Processes

Dynamic Priority Adjustment:

Dynamic Priority Adjustment in MLFQ works by assigning processes to different queues based on their CPU usage: processes with shorter CPU bursts, often I/O-bound, remain in higher-priority queues to ensure quick responsiveness, while processes with longer CPU bursts, typically CPU-bound, are gradually moved down to lower-priority queues, which balances system performance but can also create fairness issues.

Reasons I/O-Bound Processes Dominate:

1. I/O-bound processes frequently yield the CPU to perform I/O operations
2. Because they rarely exhaust their time-quantum, they retain priority
3. By contrast, CPU-bound processes consume full quanta and are pushed down to lower queues.
4. This results in I/O-bound processes monopolizing higher-priority queues, receiving faster response times.

Consequences: Starvation and Unfairness

1. CPU-bound processes may suffer starvation, waiting excessively in lower-queues.
2. The imbalance leads to unfair CPU allocation, where responsiveness for I/O-bound jobs comes at the expense of throughput for CPU-bound jobs.

Fairness-Improving Mechanisms:

Mechanism	Description	Example
Aging	Gradually increases the priority of long-waiting processes	A CPU-bound process waiting too long in a low queue is promoted upward
Priority Boosting	Periodically resets all processes to the highest queue.	Every 50ms, all processes are moved to the top queue to prevent starvation
Time Quantum Adjustment	Varying quantum lengths across queues to balance responsiveness and fairness	Longer quanta for lower queue allow CPU-bound jobs to progress meaningfully

These mechanisms help mitigate unfairness by ensuring CPU-bound processes eventually receive fair CPU time.

ii. Quantitative Fairness Evaluation:

Fairness

Fairness in CPU scheduling means equitable distribution of CPU time among processes, ensuring no process is perpetually disadvantaged. Fairness is important to balance responsiveness (for interactive tasks) and ~~throughput~~ throughput (for computational tasks) in multi-queue systems like MLFQ.

Fairness Metric

$$\text{Fairness Index} = \frac{(\sum T_i)^2}{n \times \sum T_i^2}$$

T_i : CPU time allocated to process i

n : Total number of processes

Fairness Index (FI) ranges between 0 and 1. FI = 1 indicates perfect fairness (equal CPU allocation). Lower FI value indicates greater imbalance.

Applying the Metric to a Sample MLFQ Workload

Case-1 (Default MLFQ)

Process	Type	CPU Time Allocation (T_i) (ms)
P1	I/O-bound	20
P2	I/O-bound	18
P3	CPU-bound	10
P4	CPU-bound	12

$$\Sigma T_i = 60$$

$$\Sigma T_i^2 = 20^2 + 18^2 + 10^2 + 12^2 = 968$$

$$n = 4$$

$$FI = \frac{(60)^2}{4 \times 968} = \frac{3600}{3872} \approx 0.93$$

Case-2 (Boosting + Adjusted quanta)

Process	Type	CPU Time Allocation (T_i) (ms)
P_1	I/O-bound	18
P2	I/O-bound	16
P3	CPU-bound	15
P4	CPU-bound	15

$$\Sigma T_i = 64$$

$$\Sigma T_i^2 = 18^2 + 16^2 + 15^2 + 15^2 = 1030$$

$$n = 4$$

$$FI = \frac{(64)^2}{4 \times 1030} = \frac{4096}{4120} \approx 0.99$$

Case-1 Result : I/O-bound jobs dominate higher queues; CPU-bound jobs are demoted and get less CPU

Case-2 Result : Boosting and longer queue quanta rebalance CPU share without breaking responsiveness.

Effect of Parameter changes on Fairness :

Parameter Change	Impact on Fairness	Impact on CPU Distribution
More Queues	CPU-bound jobs sink lower	Skewed toward I/O bound jobs in top queues
Longer Quanta	CPU-bound jobs progress more	More CPU time for lower queues
Frequent Aging/ Boost	Prevents starvation	Redistributes CPU time more evenly

MLFQ efficiently handles mixed workloads but favors I/O bound processes causing unfair CPU distribution. Fairness matric shows the imbalance and methods like aging, ~~priority~~ quantum tuning ~~and number of queue~~ are needed for balancing responsiveness with equitable CPU distribution.