MLFQ scheduler design

Policy

This policy employs a **round-robin** scheduling policy with **multilevel feedback** and **priority aging (usage decay)**, where:

Time unit: clock tick (10ms, based on 4.3BSD UNIX)
Number of priority queues: 128 (0, 1, 2···126,127), where 0-49 are kernel mode priorities and 50-127 are user mode priorities. (Consider to reduce it to 32 total queues later)
Quantum: 10 clock ticks based on 4.3BSD UNIX (not sure whether to have different quantums among different priority levels)
Process fields:

1. p_nice: base priority ranging from 0-19.
2. p_cpu: the accumulated and decayed CPU usage, it increments by 1 per running clock tick, every second divided by decay factor D.
3. p_usrpri: the priority a process belongs to, the higher p_usrpri, the lower the priority level.

The following methods of re-computation of process fields is modeled after UNIX System V:
After every running clock tick:
p cpu = p cpu + 1
p usrpri := PUSER + 0.5 * p cpu + p_nice

After every second:
p cpu = p cpu / D
p usrpri := PUSER + 0.5 * p cpu + p_nice

Here PUSER (to adjust the process to user mode priority) is 50, D (decay factor) is 2.

I'm not sure whether rule 4 and 5 of MLFQ must by followed under usage decay method, but rule 1-3 will be followed in this design.

Rule 1: If Priority(A) > Priority(B) run A.

Rule 2: If Priority(A) == Priority(B), run them in Round-Robin.

Rule 3: When a job enters the system, it is placed at the highest priority.

Rule 4: Once a job uses its time allotment at a given level, regardless of how many times it has given up CPU, it moves down a queue and its priority is reduced.

Rule 5: After some time period S, move all of the jobs in the system to the topmost queue.

Evaluation

The evaluation will focus on the aspects of

1. CPU utilization efficiency
2. Fairness (no starving processes)
3. Turnaround time
4. Response time
5. Throughput (I don't quite understand this metric yet)

The jobs to be tested can be classified in (assume all jobs enters in the same time)

1. All jobs are batch jobs (cpu bound)
2. All jobs are interactive jobs (IO bound)
3. A mix of the above jobs

Note

All underlined terms are better to be discussed before implementation

References

Decay-Usage Scheduling in Multiprocessors
https://www.researchgate.net/publication/234805812_Decay-Usage_Scheduling_in_Multiprocessors

Chapter 8 - Operating Systems: Three Easy Pieces
https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf