CS 343 - Operating Systems

Module-2D

CPU Scheduling Algorithms - 2

Dr. John Jose

Associate Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati

Session Outline

- ❖ Priority Scheduling
- ❖ Priority Inversion
- ❖ Multilevel Feedback Queue Scheduling
- ❖ Lottery Scheduling
- ❖ Summary of Scheduling algorithms

CPU Scheduling Algorithms

Batch Systems

- First-come first-served
- Shortest job first
- Shortest remaining Time next

Interactive Systems

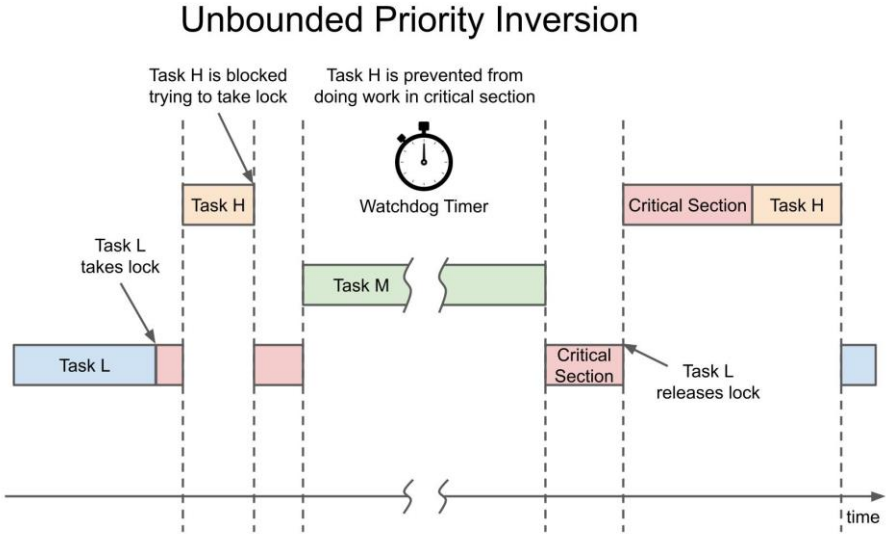
- Round-robin scheduling
- Priority scheduling
- Multi-Level queue scheduling
- Lottery scheduling
- Fair-share scheduling

Priority Scheduling

- Each process has a priority number (integer)
- Lower the integer, higher the priority
- Highest priority process is scheduled first; if equal priorities, then FCFS
- It can have 2 variants; non-preemptive and preemptive
- Arrival of a new process with a higher priority can preempt the currently running process.

Issues with Priority Scheduling – Priority Inversion

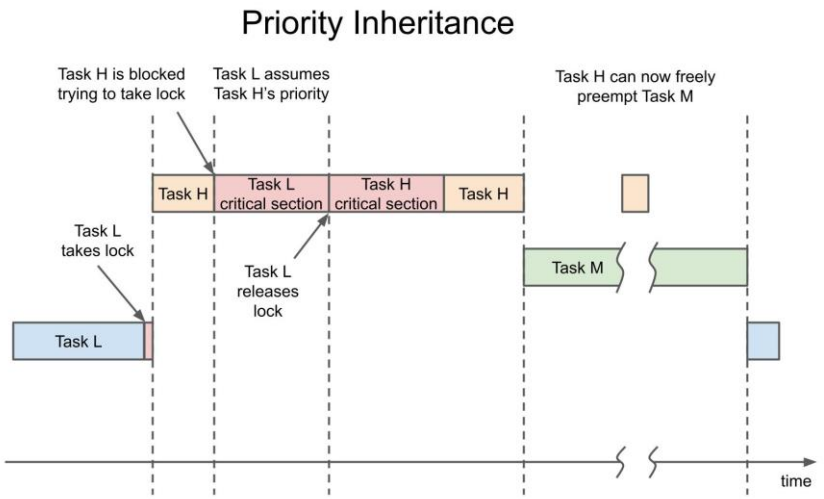
- 3 Process with varying priority: High (H), Medium (M), Low (L).
- L is running and successfully acquires a shared resource file.
- H begins by preempting L
- H tries to acquire shared resource, and blocks (held by L).
- M begins running since it has a higher priority than L, it is the highest priority ready process.



- H is in blocked state and it is in starving state.
- L progress and completes. H completes finally.

Priority Inheritance to Solve Priority Inversion

- When H gets blocked, H can donate its priority to L. (L inheriting H's priority)
- Now L is having higher priority than M.
- This enables L to run and complete/ release the resource.
- When L is finished, H becomes unblocked.
- H now being the highest priority ready process it runs, and M must wait until it is finished.

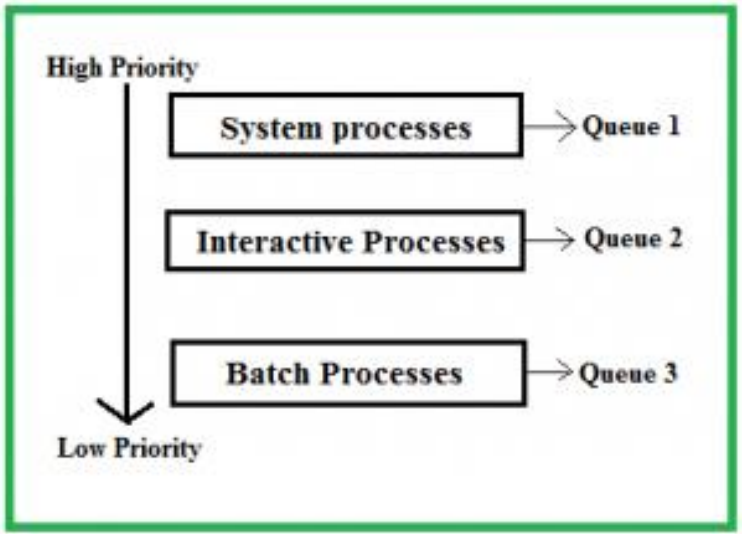


Multilevel Queue

- Ready queue is partitioned into separate queues:
  - Foreground, interactive process→ RR scheduling
  - Background, batch process→ FCFS scheduling
- A process is permanently assigned to one queue
- Each queue has its own scheduling algorithm
- Can be preemptive

Multilevel Feedback Queue Scheduling

- Scheduling must be done between the queues.
  - Fixed priority scheduling
  - Serve all from foreground then from background
  - Possibility of starvation
- Time slice
  - Each queue gets a certain amount of CPU time which it can schedule among its processes
    - i.e.: 80% Vs 20%

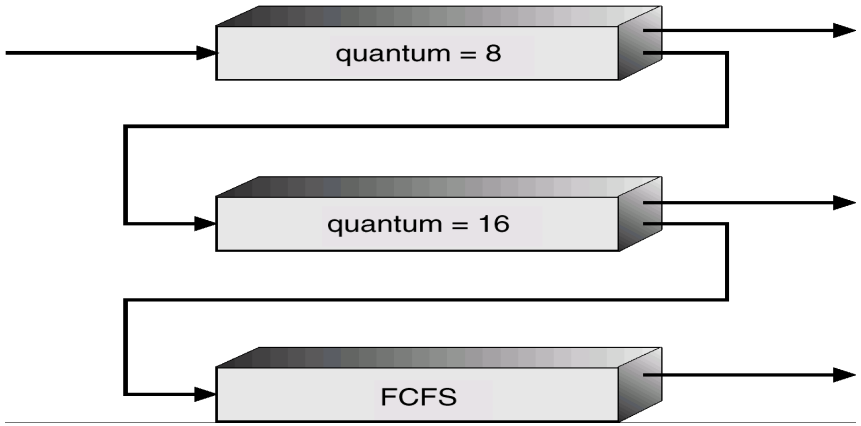


Multilevel Feedback Queue Scheduling

- **A process can move between the various queues. (Aging)**
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

Example of Multilevel Feedback Queue

- Three queues:
  - Q0 – time quantum 8 milliseconds, FCFS
  - Q1 – time quantum 16 milliseconds, FCFS
  - Q2 – FCFS



- A new job enters queue Q0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q1.
- At Q1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q2.

Lottery Scheduling

- Each job some number of lottery tickets are issued
- On each time slice, randomly pick a winning ticket
- On average, CPU time is proportional to number of tickets given to each job over time
- How to assign tickets?
  - To approximate SRTF, short-running jobs get more, long running jobs get fewer
  - To avoid starvation, every job gets at least one ticket (everyone makes progress)

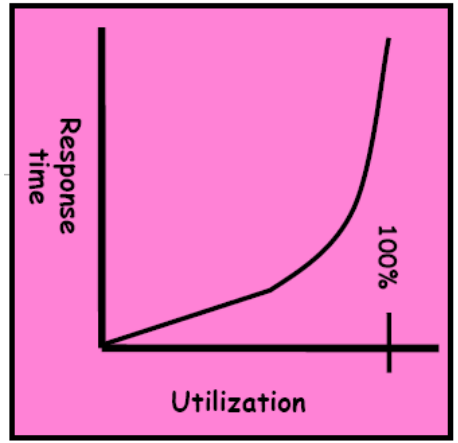
Example: Lottery Scheduling

- Assume short jobs get 10 tickets, long jobs get 1 ticket

# short jobs / # long jobs	% of CPU each short job gets	% of CPU each long job gets
1/1	91%	9%
0/2	N/A	50%
2/0	50%	N/A
10/1	9.9%	0.99%
1/10	50%	5%

Conclusion

- Scheduling: selecting a waiting process from the ready queue and allocating the CPU to it
- When do the details of the scheduling policy and fairness really matter?
  - When there aren't enough resources to go around



Conclusion

- FCFS scheduling, FIFO Run Until Done:
  - Simple, but short jobs get stuck behind long ones
- RR scheduling:
  - Give each thread a small amount of CPU time when it executes, and cycle between all ready threads
  - Better for short jobs, but poor when jobs are the same length
- SJF/SRTF:
  - Run whatever job has the least amount of computation to do / least amount of remaining computation to do
  - Optimal (average response time), but unfair; hard to predict the future

Conclusion

- Multi-Level Feedback Scheduling:
  - Multiple queues of different priorities
  - Automatic promotion/demotion of process priority to approximate SJF/SRTF
- Lottery Scheduling:
  - Give each thread a number of tickets (short tasks get more)
  - Every thread gets tickets to ensure forward progress / fairness
- Priority Scheduling:
  - Preemptive or Non-preemptive
  - Priority Inversion and Priority Inheritance

Exercise Problem-1

Consider 4 processes A, B, C, D scheduled on a CPU in round robin fashion with a time quantum of 5-time units. The processes are assumed to have arrived in the order A, B, C and D, all at time T=0. There are exactly two context switching from A to B, one context switching from B to C, and no context switching from B to D and B to A. Which one of the following is possible as CPU bursts (in time units) of these processes?

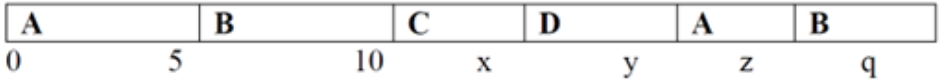
(a) A= 9, B= 10, C=12, D=5

(b) A= 8, B= 9, C=3, D=4

(c) A= 12, B= 8, C=4, D=4

(d) A= 8, B= 12, C=8, D=10

5 < CPU burst (A), CPU burst (B) <10 & 0 < CPU burst (C), CPU burst (D)<=5



A= 8, B= 9, C=3, D=4

Exercise Problem-2

Consider 4 processes P, Q, R, S scheduled in round robin fashion with a time quantum of 3-time units on a CPU. The processes are assumed to have arrived in the order P, Q, R and S, all at time T=0. There are exactly one context switching each from Q to P, R to P and S to P. There is exactly three context switching from P to Q. After three processes complete their execution, the last process will not take more than one time quantum to complete. Which one of the following is most suitable statement for these processes?

(a) Sum of CPU burst of R and S will always be larger than CPU burst of P

(b) Average turnaround time >5.5 time units

(c) Average turnaround time <5 time units

(d) R has the shortest CPU burst

Gantt chart

P	Q	R	S	P	Q	R	P	Q	P
3	3	3	1/2/3	3	3	1/2/3	3	1/2/3	1/2/3

CPU burst of the process should be as follows

CPU_burst(P)>=10 7<= CPU_burst(Q)<=9,

4<= CPU_burst(R)<=6 1<= CPU_burst(S)<=3,

Average turnaround time < 5 time units

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/