Lecture - VI

CPU Scheduling - II

Tevfik Koşar

University at Buffalo
September 13th, 2012

Roadmap

- Multilevel Feedback Queues
- Estimating CPU bursts
Multilevel Queue

- Ready queue is partitioned into separate queues: foreground (interactive) and background (batch).
- Each queue has its own scheduling algorithm:
  - foreground - RR
  - background - FCFS
- Scheduling must be done between the queues:
  - Fixed priority scheduling; (i.e., 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 amongst its processes; i.e., 80% to foreground in RR, 20% to background in FCFS.

Multilevel Queue Scheduling

[Diagram showing multilevel queue scheduling]
Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- 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:
  - $Q_0$ - RR with time quantum 8 milliseconds
  - $Q_1$ - RR time quantum 16 milliseconds
  - $Q_2$ - FCFS
- Scheduling
  - A new job enters queue $Q_0$ 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 $Q_1$.
  - At $Q_1$ job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue $Q_2$. 
How to estimate CPU burst time?
Determining Length of Next CPU Burst

- Can only estimate the length
- Can be done by using the length of previous CPU bursts, using exponential averaging

\[ \tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n. \]

1. \( t_n \) = actual length of \( n^{th} \) CPU burst
2. \( \tau_{n+1} \) = predicted value for the next CPU burst
3. \( \alpha, 0 \leq \alpha \leq 1 \)
4. Define:

Examples of Exponential Averaging

- \( \alpha = 0 \)
  - \( \tau_{n+1} = \tau_n \)
  - Recent history does not count
- \( \alpha = 1 \)
  - \( \tau_{n+1} = \alpha t_n \)
  - Only the actual last CPU burst counts

If we expand the formula, we get:

\[ \tau_{n+1} = \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \ldots \]
\[ + (1 - \alpha)^2 \alpha t_{n-j} + \ldots \]
\[ + (1 - \alpha)^{n+1} \tau_0 \]

- Since both \( \alpha \) and \( 1 - \alpha \) are less than or equal to 1, each successive term has less weight than its predecessor
Prediction of the Length of the Next CPU Burst

Exercise

Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?

a. \( \alpha = 0 \) and \( \tau_0 = 100 \text{ milliseconds} \)

b. \( \alpha = 0.99 \) and \( \tau_0 = 10 \text{ milliseconds} \)

Answer: When \( \alpha = 0 \) and \( \tau_0 = 100 \text{ milliseconds} \), the formula always makes a prediction of 100 milliseconds for the next CPU burst. When \( \alpha = 0.99 \) and \( \tau_0 = 10 \text{ milliseconds} \), the most recent behavior of the process is given much higher weight than the past history associated with the process. Consequently, the scheduling algorithm is almost memory-less, and simply predicts the length of the previous burst for the next quantum of CPU execution.
Summary

• Multilevel Feedback Queues
• Estimating CPU bursts

• Next Lecture: Project-1 Discussion

Acknowledgements

• “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne

• “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings

• “Modern Operating Systems” book and supplementary material by A. Tanenbaum

• R. Doursat and M. Yuksel from UNR