Performance Engineering of Real-Time and Embedded Systems

Real-Time Scheduling

For many, the design of real-time systems does not go beyond scheduling algorithms.

- There are a seemingly endless array of algorithms
  - ***First-Come, First-Served***
  - ***Round Robin***
  - ***Rate Monotonic***
  - ***Least Compute Time***
  - ***Shortest Completion Time***
  - ***Earliest Deadline First***
  - ***Least Slack Time***

Perhaps the most important aspect of scheduling in real-time systems is predictability.

- A ***predictable*** scheduling algorithm can mathematical show schedule feasibility
- A ***feasible*** schedule means that all tasks will run and meet their constraints
- An ***optimal*** scheduling algorithm will find a feasible schedule if it is possible
- A ***schedulability test*** validates that a scheduling algorithm will satisfy all task deadlines—usually based on utilization
- The ***utilization*** is the percentage of processor resources consumed by all tasks

Most scheduling algorithms place constraints on the tasks in the system.

- Tasks are strictly periodic
- Task deadlines are equal to their period
- Tasks are independent
- Tasks are all ready at time 0
- Tasks can be preempted at any time
- Scheduling and context switches take 0 time

- Some assumptions are relaxed by more detailed analysis

The basic task is defined by a three-tuple.

$$P_i = (c_i, p_i, d_i) : p_i = d_i$$

where

c_i is the execution time

p_i is the period

d_i is the deadline

Two scheduling algorithms from conventional systems are sometimes used.

- If the system is not tight on processor resources, it does not matter how scheduling is done
- ***First-Come, First-Served***
- ***Round-Robin*** *– preemptive*

Rate-monotonic analysis has dominated real-time scheduling

- Tasks are assigned priorities in order of period → shorter period gets higher priority
- Optimal—if there is a static priority schedule RM scheduling will satisfy task requirements
- Two simple utilization constraints test schedulability but may eliminate some RM schedulable task sets

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \quad \text{(necessary)}$$

$$U \leq n\left(2^{1/n} - 1\right) \quad \text{(sufficient but not necessary)}$$

Least compute time (LCT) is another fixed-priority scheduling algorithm

- Assign priorities in order of compute time.
- Intuition?
  - ***Tasks with shorter compute times can finish quickly so give them a higher priority***
- Not optimal

Dynamic priority schedulers adjust priorities on the fly.

- Earliest deadline first (EDF)
- Intuition?
  - ***At any scheduling point, the ready task with the earliest deadline has the highest priority***
- Optimal – if a schedule exists using dynamic priorities EDF will produce a feasible schedule
- Utilization constraint

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \quad \text{(necessary and sufficient)}$$

There are other dynamic priority scheduling algorithms.

- Shortest Completion Time (SCT) – not optimal
  - ***At any scheduling point, the task with the shortest remaining compute time has the highest priority***
- Least Slack Time (LST) – optimal
  - ***At any scheduling point, the task with the shortest time between the end of its compute time and its deadline has the highest priority***

Do not ignore the step-child scheduling algorithm.

- Cyclic Executive
  - ***Tasks are broken into executable sub-tasks***
  - ***Sub-tasks are executed in a pre-determined fixed order***
- Often used in safety critical systems
- Creating the schedule for complex systems is difficult
  - ***It is NP-hard to determine if there is a feasible schedule***