ELC 4438: Embedded System Design

Real-Time Scheduling

Liang Dong

Electrical and Computer Engineering
Baylor University

Scheduler and Schedule

- Jobs are scheduled and allocated resources according to the scheduling and resource access-control algorithms.
- A scheduler is a module that allocates processor and resources to jobs and tasks. It is defined by the scheduling and resource management algorithms it implements.
- Specifically, a job is scheduled in a time interval on a processor.
- A schedule is an assignment of all the jobs in the system on the available processors produced by the scheduler.

Scheduling Measures

- A valid schedule is a feasible schedule if every job meets its timing constraints.
- A hard real-time scheduling algorithm is optimal if using the algorithm the scheduler always produces a feasible schedule if the given set of jobs have feasible schedules.
- Minimizing maximum or average tardiness versus minimizing average absolute lateness

Scheduling Measures

- All the jobs have the same release time and deadline: The longest response time is the makespan of the schedule

- Average response time – performance measure of soft-deadline jobs

- In a system that has a mixture of jobs with hard and soft deadlines, the objective of scheduling is typically to minimize the average response time of jobs with soft deadlines while ensuring that all jobs with hard deadlines complete in time.

Scheduling Measures

- Miss rate and loss rate in soft real-time applications
- Miss rate gives the percentage of jobs that are executed but completed late.
- Loss rate gives the percentage of jobs that are discarded.
- Trade-off of miss and loss rate → invalid rate, which is the sum of miss rate and loss rate.

Approaches to Real-Time Scheduling

- Three commonly used approaches to scheduling real-time systems:

1. Clock – driven
2. (Weighted) Round – robin
3. Priority - driven

Clock-Driven Scheduling

- The job execution time intervals are made as a priori decision.
- All the parameter of real-time jobs are known, and a schedule of the jobs is computed off-line.
- Scheduling overhead during run-time can be minimized.
- Using a hardware timer: A timer is set to expire periodically without the intervention of the scheduler.

Round-Robin Scheduling

- Jobs join a FIFO queue. The job at the head of the queue executes for at most one time slice.
- If the job does not complete by the end of the time slice, it is preempted and placed at the end of the queue.
- Because the length of the slice is relatively short (typically ~10^{-3} – 10^{-2} sec), the execution of every job begins almost immediately after it becomes ready.
- Each job get 1/n th share of the process – processor-sharing algorithm

Weighted Round-Robin Scheduling

- Weighted round-robin algorithm: Rather than giving all the ready jobs equal shares of the processor, different jobs may be given different weights.

- The weight of a job refers to the fraction of processor time allocated to the job. A job with *wt* get *w* time slices in every round.

- (Weighted) round-robin scheduling is a reasonable approach for a *pipe*, since a successor job may be able to incrementally consume what is produced by a predecessor, and the two jobs can execute concurrently.

Priority-Driven Scheduling

- Priority-driven algorithms are event-driven: Scheduling decisions are made when events such as releases and completions of jobs occur.
- Priority-driven algorithms are locally greedy: The algorithms never leave any resource idle intentionally.
- Jobs ready for execution are placed in one or more queues ordered by the priorities. The priority list and rules such as whether preemption is allowed define the scheduling algorithm.
- Usually, preemptive scheduling with the ability of job migration among processors is better than non-preemptive scheduling.

Effective Release Times and Deadlines

- Assuming only one processor

- Effective Release Time: The effective release time of a job without predecessors is equal to its given release time. The effective release time of a job with predecessors is equal to the maximum value among its given release time and the effective release times of all of its predecessors.

Effective Release Times and Deadlines

- Effective Deadline: The effective deadline of a job without a successor is equal to its given deadline. The effective deadline of a job with successors is equal to the minimum value among its given deadline and the effective deadlines of all its successors.

- Calculation of effective release times and deadlines does not take into account the execution times of jobs.

Assigning Priority – EDF

- A way to assign priorities to jobs is on the basis of their deadlines.
- The earlier the deadline, the higher the priority.
- The priority-driven scheduling algorithm based on this priority assignment is called the Earliest-Deadline-First (EDF) algorithm.
- EDF algorithm is optimal when used to schedule jobs on a processor as long as preemption is allowed and jobs do not contend for resources.

Assigning Priority – LST

- The Least-Slack-Time-First (LST) algorithm assigns priorities to jobs based on their slacks: the smaller the slack, the higher the priority.
- At any time t, the slack (or laxity) of a job with deadline at d is equal to d-t minus the time required to complete the remaining portion of the job.
- LST algorithm is also optimal for scheduling preemptive jobs on one processor.
- Note: The EDF and the LST algorithms are optimal only when preemption is allowed.

Advantages of Priority-Driven Scheduling

- Priority-driven scheduling is easy to implement. It does not require the information on the release times and execution times of the jobs a priori.

- The run-time overhead due to maintaining a priority queue of ready jobs can be made small.

On-Line Scheduling

- The priority-driven algorithms are on-line scheduling algorithms.
- The scheduler of on-line scheduling makes each decision without knowledge about the jobs that are released in the future.
- The parameters of each job become known to the on-line scheduler only after the job is released.
- On-line scheduling is the only option in a system whose future workload is unpredictable.

Disadvantages of Priority-Driven Scheduling

- However, the timing behavior of a priority-driven system is nondeterministic.

- It is difficult to validate that all jobs scheduled in a priority-driven manner meet their deadlines when the job parameters vary.

Assumptions of Priority-Driven Scheduling

- Every job is ready for execution as soon as it is released, and can be preempted at any time.
- Scheduling decisions are made immediately upon job releases and completions.
- The context switch overhead is negligibly small compared with execution times.
- The number of priority levels is unlimited.

Scheduling on Uniprocessor Systems

- In a static system, all the tasks are partitioned into subsystems. Each subsystem is assigned to a processor, and tasks on each processor are scheduled by themselves.

- In a dynamic system, jobs ready for execution are placed in one common priority queue and dispatched to processors for execution as the processors become available.

- Most hard real-time systems built and in use to date are static. In the case when tasks in a static system are independent, we consider scheduling jobs to a single processor.

Fixed versus Dynamic Priority Algorithms

- A fixed-priority algorithm assigns the same priority to all the jobs in each task.
- A dynamic-priority algorithm assigns different priorities to the individual jobs in each task. By dynamic, we mean task-level dynamic and job level fixed.
- Most real-time scheduling algorithms of practical interests assign job-level fixed priorities.

Fixed-Priority Algorithms

- Rate-monotonic (RM) algorithm: It assigns priorities to tasks based on their periods: the shorter the period, the higher the priority. Hence, the higher the rate, the higher the priority.

- Deadline-monotonic (DM) algorithm: It assigns priorities to jobs according to their relative deadlines: the shorter the relative deadline, the higher the priority.

DM versus RM Algorithms

- When the relative deadlines are arbitrary, the DM algorithm performs better in the sense that it can sometimes produce a feasible schedule when the RM algorithm fails, while the RM algorithm always fails when the DM algorithm fails.

Dynamic-Priority Algorithms

- Jobs are assigned different priority-level in different tasks.

- Once a job is placed in the ready job queue according to the priority assigned to it, its order with respect to other jobs in the queue remains fixed.

EDF Algorithm

- Earliest-Deadline-First (EDF) algorithm assigns priorities to individual jobs in the tasks according to their absolute deadlines. The earlier the deadline, the higher the priority. It is a dynamic-priority algorithm.

- This algorithm is optimal when used to schedule jobs on a processor as long as preemption is allowed and jobs do not contend for resources.

- Definition of “optimal”: can produce a feasible schedule of a set of jobs with arbitrary release times and deadlines on a processor if a feasible schedule exists.

LST Algorithm

- Least-Slack-Time-First (LST) algorithm, a.k.a. Minimum-Laxity-First (MLF) algorithm, assigns priorities to jobs based on their slacks: the smaller the slack, the higher the priority.

- At any time t, the slack (or laxity) of a job with deadline at d is equal to d – t minus the time required to complete the remaining portion of the job.

- EDF and LST algorithms are optimal only when preemption is allowed.

Overloaded Systems

- The system is overloaded if the jobs offered to the scheduler cannot be feasibly scheduled by any scheduler.
- During an overload, some jobs must be discarded in order to allow other jobs to complete in time.
- EDF and LST algorithms are optimal under the condition that the jobs are preemptive, there is only one processor, and the processor is not overloaded.
- EDF and LST algorithms performance poorly when the system is overloaded.

Maximum Schedulable Utilization

- A system of independent, preemptive tasks with relative deadlines equal to their respective periods can be feasibly scheduled on one processor if and only if its total utilization is equal to or less than 1.

- A system of simply periodic, independent, preemptive tasks whose relative deadlines are equal to or larger than their periods is schedulable on one processor according to the RM algorithm if and only if its total utilization is equal to or less than 1.

Maximum Schedulable Utilization

- A system of independent, preemptive periodic tasks that are in phase and have relative deadlines equal to or less than their respective periods can be feasibly scheduled on one processor according to the DM algorithm whenever it can be feasibly scheduled according to any fixed-priority algorithm.