BIOINFO Genetic Programming
Volume 1, Issue 1, 2011, pp-05-07
Available online at: http://www.bioinfo.in/contents.php?id=280

Real-Time System: Best Job of using advanced Task Scheduling

H.S. Fadewar[1] S.V. Kakade[2] and Desai A.P. [3]

[1] *Sinhgad Insitute of Management and Computer Application, Narhe, Pune.*
[2] *Swami Vivekanand Mahavidyalaya, Udgir.*
[3] *Rajiv Gandhi College of Computer Science, Vidyut Nagar, Nanded*

Abstract :

In adaptive soft real time system an acceptable deadline misses and delays are tolerable. The main objective of this work is to design and development of efficient preprocess scheduler that will select the algorithm which is best suited for the particular problem in the real time environment.

1. INTRODUCTION

Task scheduling is the main activity in the design of Real-Time System (RTS). It assures both functionality and safety of such systems. RTS can be modeled as a set of periodic tasks that must be completed before specific deadlines.

The scheduling algorithms used in a particular application can have a significant impact on the functionality of the real-time system. One effect is to accumulate the a periodic tasks at a point in time in an overloaded system. In this situation the scheduler may not be able to meet all of the a periodic and periodic tasks deadlines [1].

Every algorithm has a specific set of type of input and produces the output corresponding to the input pattern. In this project the objective is to find such algorithms and then design a scheduler that will select the particular algorithm depending upon the given input pattern and apply dynamically to the workload.

2. SCHEDULING MECHANISMS

A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. Another main reason is the need for processes to perform I/O operations in the normal course of computation. Since I/O operations ordinarily require orders of magnitude more time to complete than do CPU instructions, multiprograming systems allocate the CPU to another process whenever a process invokes an I/O operation .Here we are using the scheduling to schedule the process for execution [2,3].

3. TASK SCHEDULING

Most RTOSs do their scheduling of tasks using a scheme called “priority-based preemptive scheduling.” Each task in a software application must be assigned a priority, with higher priority values representing the need for quicker responsiveness. Very quick responsiveness is made possible by the “preemptive” nature of the task scheduling. “Preemptive” means that the scheduler is allowed to stop any task at any point in its execution, if it determines that another task needs to run immediately [2].

The basic rule that governs priority-based preemptive scheduling is that at every moment in time, “The Highest Priority Task that is Ready to Run, will be the Task that Must be Running.” In other words, if both a low-priority task and a higher-priority task are ready to run, the scheduler will allow the higher-priority task to run first. The low-priority task will only get to run after the higher-priority task has finished with its current work [David Kalinsky, 2004]. What if a low-priority task has already begun to run, and then a higher-priority task becomes ready? This might occur because of an external world trigger such as a switch closing. A priority-based preemptive scheduler will behave as follows:

It will allow the low-priority task to complete the current assembly-language instruction that it is executing. (But it won’t allow it to complete an entire line of high-level language code; nor will it allow it to continue running until the next clock tick.) It will then immediately stop the execution of the low-priority task, and allow the higher-priority task to run. After the higher-priority task has finished its current work, the low-priority task will be allowed to continue running. This is shown in Figure 1, where the higher-priority task is called “Mid-Priority Task.” Of course, while the mid-priority task is running, an even higher-priority task might become ready. This is represented in Figure 1 by “Trigger_2” causing the “High-Priority Task” to become ready. In that case, the running task (“Mid-Priority Task”) would be preempted to allow the high-priority task to run. When the high-priority task has finished its current work, the mid-priority task would be allowed to continue. And after both the high-priority task and the mid-priority task complete their work, the low-priority task would be allowed to continue running. This situation might be called “nested preemption.” [4]

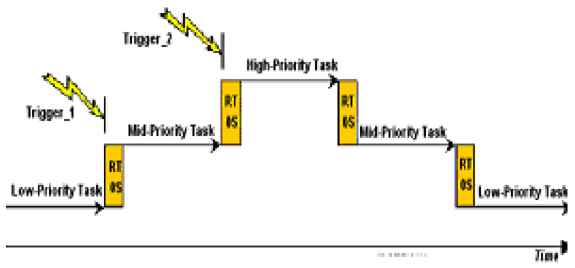

Figure 1: Timeline for Priority-based Preemptive Scheduling Examples.

Each time the priority-based preemptive scheduler is alerted by an external world trigger (such as a switch closing) or a software trigger (such as a message arrival), it must go through the following 5 steps:

- Determine whether the currently running task should continue to run. If not
- Determine which task should run next.
- Savethe environment of the task that was stopped (so it can continue later).
- Set up the running environment of the task that will run next.
- Allow this task to run.

These 5 steps together are called “*task switching*.”

3.1. Advanced Task Scheduler :

Advanced Task Scheduler - is a multifunctional task scheduler, which allows launching programs, scripts and batch files, opening documents and Internet pages, displaying popup reminders, playing sounds, sending messages, shutting down and restarting computer, stopping running processes, establishing and closing dial-up connections - automatically. Advanced Task Scheduler offers full set of scheduling tools that allow running scheduled tasks automatically once, minutely, hourly, daily, monthly, yearly, in specified period of time after starting the computer or by such events as hot key, computer idle, dial-up connection established/terminated, user logged on/ logged off, program started/stopped and so on. Advanced Task Scheduler can automate many of your routine tasks. Automatic launching of programs with flexible set of planning tools will set you free from having to wait until some time to run needed applications. Popup reminders let you not forget important things that you were planning to get done. Automatic shutdown feature allows leaving the computer running while being sure that it will be shut down at specified hour. The capability of stopping processes at specified hour makes it possible not only starting programs automatically but stopping programs automatically as well. Automatic opening and closing dial-up connections allows both establishing and closing dial-up connections at extremely accurate moment of time[5].

Advanced Task Scheduler places its icon to system tray. This provides access to all features of the scheduler via the popup menu, which appears by right-clicking on the icon. Advanced Task Scheduler can also be restored by pressing the hotkey.

Advanced Task Scheduler can be started as a Windows Service and work in the background so it will not take a place on the desktop but all scheduled tasks will work normally. This feature allows run Advanced Task Scheduler even when no user is logged on[5].

Advanced Task Scheduler can record all executed tasks to the log file or send them to an email address. With this log, you will always be informed which task and at what time was executed. The log file can be printed out at any time.

4. Task Types :

A task is a sequential program that is invoked for execution by the occurrence of a particular event. Tasks may be either periodic, aperiodic or sporadic in nature. A periodic task is characterized by a release time, a deadline, and a period. The release time is the time at which the task is ready to execute, the deadline is the time by which the task must complete execution, and the period is the exact spacing between successive invocations of the task. When the release time of the task is specified before it is scheduled, the task is called a concrete periodic task. When release times are arbitrary, a task is invoked periodically after its first release[6]. A periodic tasks have soft or no deadlines. Sporadic tasks are tasks that may enter and leave the system at any time. Sporadic tasks are characterized by a release time, a deadline, and a period. For a sporadic task, the period represents the minimum time after which the invocation of the next task occurs. When release times are specified in advance, scheduling decisions can be made off-line or statically. When release times are arbitrary, scheduling decisions are made on-line. Figure. 2 illustrate this classification [Litoiu & Tadei, 2001].

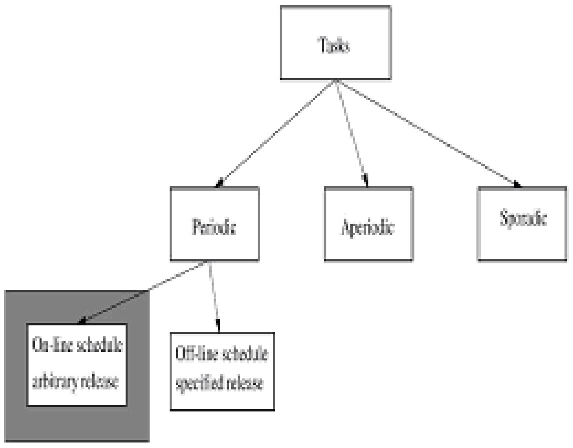
Tasks
Periodic
Aperiodic
Sporadic
On-line schedule arbitrary release
Off-line schedule specified release

Fig. 2. Classification of the different types of tasks.

4.1 Real-Time Task Model :

A real-time application is specified by means of a set of tasks. Real-time tasks are the basic executable entities that are scheduled; they may be periodic or a periodic, and have hard (late data are bad data) or soft (late data may still be good data) real-time constraints. The quality of scheduling depends on the exactness of these parameters, so their determination is an important aspect of real-time design[7].

- r, task release time, i.e. the triggering time of the task execution request.
- C, task worst-case computation time, when the processor is fully allocated to it.
- D, task relative deadline, i.e. the maximum acceptable delay for its processing.
- T, task period (valid only for periodic tasks).
- When the task has hard real-time constraints, the relative deadline allows computation of the absolute deadline d = r + D. Transgression of the absolute deadline causes a timing fault. Also, when tasks are allowed to access shared resources, their access needs to be controlled in order to maintain data consistency.

5. CONCLUSION

This work is to design and development of efficient preprocess scheduler that will select the algorithm which is best suited for the particular problem in the real time environment.

References

[1] [Churnetski, 2003] Kevin Churnetski "A comparison of real-time scheduling algorithms using visualization of tasks and evaluation of real-time extensions to Linux " Computer Science-RIT in 2003.

[2] [Lipari at. Al., 2003] Giuseppe Lipari, Enrico Bini, Gerhard Folher, "A Framework for Composing Real-Time Schedulers", Elsevier Science B. V. in 2003.

[3] [Ahmad at. Al., 2003] Idawaty Ahmad, S.Shamala, M.Othman and Muhammad Fauzan Othman "A Preemptive Utility Accrual Scheduling Algorithm for Adaptive Real Time System".

[4] [Yu at. al., 2003] Haobo Yu, Andreas Gerstlauer, Daniel Gajski " RTOS Scheduling in Transaction Level Models " CECS Technical Report 20/03/2003.

[5] [Mercer, 2002] Clifford W. Mercer "An Introduction to Real-Time Operating Systems: Scheduling Theory" , School of Computer Science Carnegie Mellon University Pittsburgh, Pennsylvania 15213.

[6] [Litoiu & Tadei, 2001] Marin Litoiu, Roberto Tadei, "Real-time task scheduling with fuzzy deadlines and processing times", Elsevier Science B. V. in 2001.

[7] [Litoiu & Tadei, 2001] Marin Litoiu, Roberto Tadei, "Fuzzy scheduling with application to Real-time system", Elsevier Science B. V. in 2001.