INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

A REVIEW OF DEFERRED PREEMPTIVE SCHEDULING FOR REAL TIME TRANSACTIONS

MRS. R. R. LONKAR[1], PROF. S. A. BHURA [2]

1. Assistant Professor, Vidyabharati Maha vidyalaya, Amravati.
2. Associate professor, HOD, DCPE HVPM, Amravati.

Accepted Date: 05/03/2015; Published Date: 01/05/2015

Abstract: Real time system had become important part of industry. Systems that are subject to a "real-time constraint"—i.e., operational deadlines from event to system response. Real-time programs must execute within strict constraints on response time. In Real-time systems preemption is consider as key factor for scheduling it allows operating system to immedialty allocate the processor requiring urgent service. On the whole, deadline is one of the important aspects of real time systems which have to be carefully understood when we are working with transaction. The question whether preemptive algorithms are better than non-preemptive ones for scheduling a set of real-time tasks has been debated for a long time in the research community, each approach has advantages and disadvantages, and no one dominates the other when both predictability and efficiency have to be taken into account in the system design. Recently, limited preemption models have been proposed as a viable alternative between the two extreme cases of fully preemptive and non-preemptive scheduling. This paper present survey of existing approach for reducing the preemption and compare them under different matrices providing both quantitave and qualitative performance evaluation.

Keywords: Real-time systems, Non-preemptive regions, Limited-preemptive Scheduling, Deferred preemptive scheduling, Feasibility analysis, longest non-preemptive interval.

Corresponding Author: MRS. R. R. LONKAR

Access Online On:

www.ijpret.com

How to Cite This Article:

R. R. Lonkar, IJPRET, 2015; Volume 3 (9): 210-219

PAPER-QR CODE

INTRODUCTION

Real time system had become important part of industry. Systems that are subject to a "real-time constraint"—i.e., operational deadlines from event to system response. Real-time programs must execute within strict constraints on response time. A real-time system may be one where its application can be considered (within context) to be mission critical. The anti-lock brakes on a car are a simple example of a real-time computing system — the real-time constraint in this system is the time in which the brakes must be released to prevent the wheel from locking. Real-time computations can be said to have *failed* if they are not completed before their deadline, where their deadline is relative to an event. A real-time deadline must be met, regardless of system load.

Real-time systems typically employ a collection of application tasks or threads that must complete their work within real-time constraints. Hence, scheduling real-time tasks is complicated work since it requires pre-computed information about the task's timing properties. Although we know the system's timing properties, scheduling tasks is highly concentrated mental work. Moreover the system designer is responsible to see that system meets real time constraints. For guaranteeing feasibility of generated scheduling, several static and dynamic priority scheduling algorithms and corresponding schedulability tests are introduced such as Rate-Monotonic (RM), Deadline-Monotonic (DM) and Earliest Deadline First (EDF).

In Real-time systems preemption is consider as key factor for scheduling it allows operating system to immedialty allocate the processor requiring urgent service. On the whole, deadline is one of the important aspects of real time systems which have to be carefully understood when we are working with transaction. However, Fully preemptive scheduler produces many unnecessary preemption. To reduce runtime overhead due to preemption and preserve scheduability of task set we will explore "Deferred Preemptive Scheduling,"

The Deferred Preemptive Scheduling (DPS) deferred is the appropriate word because preemption is postpone for given amount of time rather than moved to specific position in the code. Depending upon how non-preemptive regions are implemented we have been consider different models ie. Floating model and Activation triggered model.

LITURATURE REVIEW

M. Bertogna and S. Baruah[1] presented “Limited preemption EDF scheduling of sporadic task systems,” proposes a limited-preemption scheduling technique will be presented to join the benefits of both preemptive and non-preemptive scheduling. They have designed and analyzed an EDF-based limited-preemption algorithm for scheduling sporadic task systems upon uniprocessor platforms. This limited-preemption algorithm combines beneficial features from both preemptive and non-preemptive EDF. The run-time behavior of preemptive EDF scheduling will be improved, avoiding preemptions that are not needed to satisfy the schedulability of the system. As with preemptive EDF, limited-preemption EDF has low schedulability overhead and is amenable to efficient feasibility analysis; as with non-preemptive EDF, it offers low run-time overhead and is typically able to offer efficient support for mutual exclusion.

To evaluate the effectiveness of the proposed algorithm, they performed a series of experiments with various different kinds of randomly generated task sets. They simulated the scheduling of these task sets for a sufficiently large time-interval using EDF and limited-preemption EDF, counting the total number of preemptions for each of the considered algorithms.

R. J. Bril, J. J. Lukkien, and W. F. J. Verhaegh[2] in their paper, “Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited” studied the problem of computing a preemptive schedule of equal-length jobs with given release times, deadlines and weights. Our goal is to maximize the weighted throughput, which is the total weight of completed job. They provides a revised analysis, resolving the problems with the existing approaches. The analysis is based on known concepts of critical instant and busy period for FPPS. To accommodate for our scheduling model for FPDS, we need to slightly modify existing definitions of these concepts.

Giorgio C. Buttazzo , Marko Bertogna, and Gang Yao[3] proposes “Limited preemptive Scheduling for Real-Time Systems - A Survey” presents a survey of the limited preemption approaches for reducing preemptions and compares them under different metrics, providing both qualitative and quantitative performance evaluations. They presented a survey of limited preemptive scheduling algorithms, as methods for increasing the predictability and efficiency of real-time systems. The most relevant result that clearly emerges from the experiments is that, under fixed priority scheduling, any of the considered algorithms dominates both fully preemptive and non-preemptive scheduling, even when preemption cost is neglected.

Each specific algorithm preemption threshold mechanism has a simple and intuitive interface and can be implemented by introducing a low runtime overhead; however, preemption cost cannot be easily estimated, since the position of each preemption, as well as the overall number of preemptions for each task, cannot be determined offline. Using deferred preemptions, the number of preemptions for each task can be better estimated, but still the position of each preemption cannot be determined off-line. Moreover, simulation experiments clearly show that the FPP(Fixed preemption points) algorithm is the one generating less preemptions and higher schedulability ratios for any task set parameter configurations.

C. Liu and J. Layland[4] proposes "Scheduling algorithms for multiprogramming in a hard-real-time environment" has discussed some of the problems associated with multiprogramming in a hard-real-time environment typified by process control and monitoring, using some assumptions which seem to characterize that application. A scheduling algorithm which assigns priorities to tasks in a monotonic relation to their request rates was shown to be optimum among the class of all fixed priority scheduling algorithms. The least upper bound to processor utilization factor for this algorithm is on the order of 70 percent for large task sets. The dynamic deadline driven scheduling algorithm was then shown to be globally optimum and capable of achieving full processor utilization. A combination of the two scheduling algorithms was then discussed; this appears to provide most of the benefits of the deadline driven scheduling algorithm, and yet may be readily implemented in existing computers.

Gang Yao, Giorgio Buttazzo and Marko Bertogna[5] in their paper **"**Feasibility analysis under fixed priority scheduling with limited preemptions" considered the problem of analyzing the feasibility of a task set with limited preemptions under fixed priority scheduling. They presents the schedulability analysis of real-time tasks with nonpreemptive regions, under fixed priority assignments. In particular, two different preemption models are considered: the floating and the fixed preemption point model. Under each model, the feasibility analysis is addressed by deriving simple and effective schedulability tests, as well as an algorithm for computing the maximum length of the non-preemptive regions for each task. Finally, simulation experiments are presented to compare the two models in terms of schedulability.

Harini Ramprasad and Frank Mueller [6] in their paper "Bounding Worst Case Response time With Non Preemptive Region" proposed framework for safe and tight bounds of data cache related preemption delay. WCET and Worst Case Response times, not just for homogeneous task for fully Preemptive and Fully Non Preemptive System but for the task with Non-Preemptive region. By obtaining response time bound for the task set where some task have Non Preemptive Region and comparing the result with equivalent task set where only those task with Non Preemptive region are schedule Non Preemptively altogether. we show that for some task set scheduablity is improve without significantly affecting response time of the task partially Non Preemptive task as opposed to the Fully Non Preemptive task. This is the first framework that bounds D- CRPD and response time for the task with Non Preemptive Region.

Bipasa Chattopadhyay and Sanjoy Baruah[7] in their paper "Limited-preemption Scheduling on Multiprocessors" have obtained a pseudo-polynomial time schedulability test for limited-preemption scheduling under GEDF(Global Earliest Deadline First). They have shown how to apply this schedulability test to a multiprocessor multi-GPU system with non-preemptive busy-waiting. They have also indicated how further analysis may provide better analytical results for the multi-GPU system model under consideration. A comparison of analytical results under different multi-GPU system models is merited.

Fengxiang Zhang, and Alan Burns[8] in their paper "Schedulability Analysis for Real-Time systems with EDF Scheduling" propose new results on necessary and sufficient schedulability analysis for EDF scheduling; the new results reduce, exponentially, the calculation times, in all situations, for schedulable task sets, and in most situations, for unschedulable task sets. For example, a 16-task system that in the previous analysis had to check 858,331 points (deadlines) can, with the new analysis, be checked at just 12 points. There are no restrictions on the new results: each task can be periodic or sporadic, with relative deadline, which can be less than, equal to, or greater than its period, and task parameters can range over many orders of magnitude.

PROBLEM DEFINITION

Whether preemptive algorithms are better than non-preemptive ones for scheduling a set of real-time tasks has been debated for a long time in the research community. In fact, limited preemption s have been consider as a viable alternative between the two extreme cases of fully preemptive and nonpreemptive scheduling. A hybrid limited-preemption real-time scheduling algorithm where necessary for maintaining feasibility, but attempts to avoid unnecessary preemptions during run-time. The positive effects of this approach are not limited to a reduced run-time overhead, but will be extended as well to a simplified handling of shared resources.

In our project survey of limited preemptive scheduling algorithms, as methods for increasing the predictability and efficiency of real-time systems for reducing preemptions and compares them under different metrics, providing both qualitative and quantitative performance evaluations

PROPOSED SYSTEM

In this project, first various real time CPU scheduling algorithms will be studied and analyzed. Here we will include the concepts related to CPU scheduling. We will study various real-time transaction scheduling. These algorithms will be implemented and compared them with other existing algorithms and their performance will be evaluated.

Here, we will describe deffered preemptive scheduling approaches proposed in the literature to handle limited preemptive scheduling. We describes the task model and the terminology adopted which presents the schedulability analysis for the non-preemptive task model. Analysis of preemption model reports and discusses some simulation results and states our conclusions.

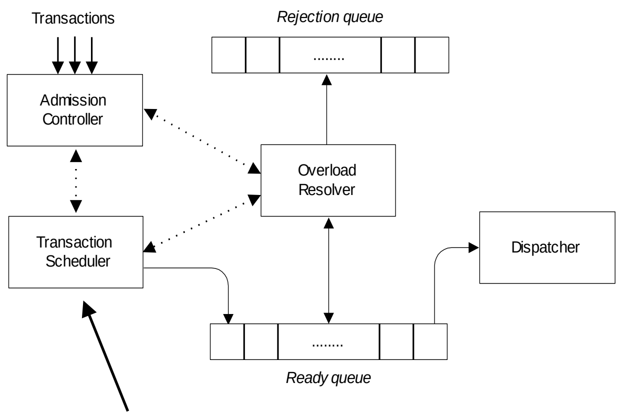
In real-time computing scenarios, we are concerned with logical as well as temporal accuracy of the results. While developing algorithms or performing schedulability analysis, preemption overheads are usually ignored even, though it's an operation that consumes finite amount of time. Preemption allows higher priority tasks to halt the lower priority tasks, though this need not be necessarily required to meet task deadlines. Our concern is not to get the results as soon as possible but soon enough to just meet the deadlines of all the tasks.

Deffered preemptive Scheduling (DPS), each task define maximum interval of time qi in which it can execute nonpreemptively. Depending upon implemention nonpreemptive region start after invocation of system call inserted at the beginning of nonpreemptive region(In Floating Model) or triggered by higher priority task(In activation Triggered Model).When Using Deffered Preemptive method we need to Calculate Longest non-Preemptive interval for each task that still preserved the task set scheduability. In presence of non-preemtive intervals task is blocked when at its arrival by, lower priority task running in nonpreemptive mode. Each task is block at most once by lower priority task, Bi is equal to Longest non-preemptive interval belonging to the lower priority task.we need to calculate blocking factor.Deffered Preemption can be considered as a trade-off between fully preemptive and fully non-preemptive scheduling. Using deffered preemption ,number of preemption for each task is better estimated;

To better appreciate the importance of limited preemptive scheduling and to better understand the difference among the limited preemptive approaches here, the real-time scheduling algorithms like Rate-Monotonic (RM), Deadline-Monotonic (DM) and Earliest Deadline First (EDF) schedule task set , it results to be unschedulable both in fully preemptive and in fully non-preemptive mode, but it can be schedulable by all limited preemptive approaches while guaranteeing their respective deadlines .

Depending upon these existing real-time scheduling algorithms, we will try to design our new algorithms for deffered Earliest Deadline First for CPU scheduling while considering above concepts. Then their performance will be evaluated.

Flow of proposed system design



Deffered Preemptive Scheduling (PTS)

According to this method, each task ti define maximum interval of time qi in which it can execute nonpreemptively. Depending upon implementations nonpreemptive region start after invocation of system call inserted at the beginning of nonpreemptive region(In Floating Model) or triggered by higher priority task(In activation Triggered Model).When Using Deferred Preemptive method we need to Calculate Longest non-Preemptive interval for each task that still preserved the task set scheduability. In presence of nonpreemtive intervals task is blocked when at its arrival by, lower priority task running innonpreemptive mode. Each task is block at most once by lower priority task, Bi is equal to longest non-preemptive interval belonging to the lower priority task. We need to calculate blocking factor. Deferred Preemption can be considered as a trade-off between fully preemptive and fully non-preemptive scheduling. Using deferred preemption number of preemption for each task is better estimated;

Feasibility study

In the presence of non-preemptive intervals, a task can be blocked when, at its arrival, a lower priority task is running in non-preemptive mode. Since each task can be blocked at most once by a single lower priority task, Bi is equal to the longest non-preemptive interval belonging to tasks with lower priority. That is, Bi= maxj:Pj <Pi{q j − 1}

Then, schedulability can be checked through the response time analysis, by Equation shown below, Note that such an analysis is exact under the floating model. In fact, since non-preemptive regions can have an arbitrary duration no greater than qi, the worst-case interference on τi can actually occur, assuming that τi can be preempted an epsilon before its completion

$$\begin{cases} R_i^{(0)} = B_i + C_i \\ R_i^{(s)} = B_i + C_i + \sum_{h:P_h > P_i} \left\lceil \frac{R_i^{(s-1)}}{T_h} \right\rceil C_h. \end{cases}$$

8. Longest non preemptive interval

When using the deferred preemption method, an interesting problem is to find the longest non-preemptive interval Qi for each task τi that can still preserve the task set schedulability. More precisely, the problem can be stated as follows:

Given a set of n preemptive periodic tasks that is feasible under a fixed priority assignment, find the longest non-preemptive interval of length Qi for each task τi , so that τi can continue to execute for Qi units of time in non-preemptive mode, without violating the schedulability of the original system.

The longest non-preemptive interval Qi of task τi that preserves feasibility can be computed as

$$Q_i = \begin{cases} \infty & if\, i = 1 \\ \min\{Q_{i-1}, \beta_{i-1}\} & otherwise \end{cases}$$

Proof. The theorem can be proved by noting that

$$\min_{k:P_k > P_i} \{\beta_k\} = \min\{\min_{k:P_k > P_{i-1}} \{\beta_k\}, \beta_{i-1}\}$$

Since from the equation

$$Q_{i-1} = \min_{k:P_k > P_{i-1}} \{\beta_k\}$$

We have that

$$Q_i = \min\{Q_{i-1}, \beta_{i-1}\}$$

Which proves the theorem

Note that, in order to apply Theorem , Qi is not constrained to be less than or equal to Ci , but a value of Qi greater than Ci means that τi can be fully executed in non-preemptive mode. The algorithm for computing the longest non-preemptive intervals is illustrated below

```
Algorithm: Compute the Longest Non-Preemptive Intervals
Input: A task set $\mathcal{T}$ with $\{C_i, T_i, D_i, P_i\}, \forall \tau_i \in \mathcal{T}$
Output: $Q_i, \forall \tau_i \in \mathcal{T}$
// Assumes $\mathcal{T}$ is preemptively feasible and $D_i \leq T_i$
(1) begin
(2)     $\beta_1 = D_1 - C_1$;
(3)     $Q_1 = \infty$;
(4)     for ($i := 2$ to $n$) do
(5)         $Q_i = \min\{Q_{i-1}, \beta_{i-1}\}$;
(6)         Compute $\beta_i$ using Equation
(7)     end
(8) end
```

Fig. shows Algorithm for computing Longest Non Preemptive Intervals

$$\beta_i = T_i \left[U_{lub}(i) - \sum_{h:P_h > P_i} \frac{C_h}{T_h} \right].$$

$$\beta_i = \max_{t \in \mathcal{TS}(\tau_i)} \{t - W_i(t)\}.$$

CONCLUSION

Limited preemptive scheduling algorithms, as methods for increasing the predictability and efficiency of real-time systems. Limited Preemptive help in improving the I/O scheduling. Specific algorithm for limiting preemptions has advantages and disadvantages. Using deferred preemptions, the number of preemptions for each task can be better estimated, but still the position of each preemption cannot be determined offline. Finally our aim is to survey the exixting approaches for reducing pre-emption and compare them under different metricts, provding both quantitative and qualitative performance evaluation.

REFERENCES

1. Bertogna and Baruah,2010]M. Bertogna and S.Baruah, "Limited preemption EDF scheduling of sporadic task systems," IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pp. 579–591, 2010.

2. Bril, Lukkien and Verhaegh ,2007]R. J. Bril, J. J. Lukkien, and W.F.J.Verhaegh, "Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited" in Proc. Of the 19th Euromicro Conf. on Real-Time Systems (ECRTS'07), Pisa, Italy, July 4-6, 2007, pp. 269–279.

3. Buttazzo, Bertogna and Yao,2013] Giorgio C.Buttazzo, Marko Bertogna and Gang Yao, "Limited Preemptive Scheduling for Real-Time System A Survey", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS,VOL. 9, NO. 1, FEBRUARY 013.

4. Liu and Layl and, 1973] C. Liu and J. Layland "Scheduling algorithms for multiprogramming in a hard-real-time environment," Journal of the Association for computing Machinery, vol. 20, no. 1, pp. 46–61, January 1973.technol.

5. Yao, Buttazzo and Bertogna,2013]Gang Yao, Giorgio Buttazzo and Marko Bertogna, **"**Feasibility analysis under fixed priority scheduling with limited preemptions". Published online: 14 January2011, Springer Science Business Media, LLC2011 Real-Time Syst (2011) 47: 198–223. DOI 10.1007/s11241-010-9113-6.

6. H.Ramprasad and F.Mueller "Bounding worst case response time for the task with non preemptiveregion", Real Time Embeded Appl.symp(RTAS 08) St Louis Mo Apr 22-24 2008 pp 58-67.

7. Chattopadhyay and Baruah, 2014] Bipasa Chattopadhyay and Sanjoy Baruah, "Limited preemption Scheduling on Multiprocessors", Department of Computer Science The University of North Carolina at Chapel Hill, RTNS 2014, October 8 - 10 2014, Versailles.

8. Zhang and Burns,2009] Fengxiang Zhang, and Alan Burns "Schedulability Analysis for Real-Time Systems with EDF Scheduling", IEEE TRANSACTIONS ON COMPUTERSVOL. 58, NO. XX, XX 2009.