AN ADAPTIVE SCHEDULING ALGORITHM FOR MIXED IN-DEPENDED TASKSET ON WEAKLY HARD REAL-TIME SYSTEMS

RULA MREISHEH PHD*
DR. MOHAMMED HEJAZEA**

*Student in Computer Engineering and Automatic Control Department, Faculty of Mechanical and Electrical Engineering - Tishreen University, Syria
**Assistant Professor-Computer Engineering and Automatic Control Department- Faculty of Mechanical and Electrical Engineering- Tishreen University, Syria

Abstract
This paper proposes an adaptive scheduling algorithm for multiple types of real-time tasks (periodic soft real-time tasks, sporadic soft real-time tasks and a periodic hard real-time tasks) on identical multiprocessing environment. With the practical application of complicated environment that called complex real-time system (soft and hard real time tasks), we propose an improved scheduling algorithm in this paper, which modifies EDH Salgorithm (Earliest Deadline and Highest-priority Split) to be more affective in scheduling mixed tasks with soft and hard constrains in a weakly hard real time systems. This algorithm can effectively improve the success rate of the hard tasks and the entire mixed task set scheduling by decreasing the preemptions counts, context switches and tasks average waiting time in queues. The results in simulation experiments show that the performance of the proposed algorithm scheduling on this complex real time system has obvious advantages over that of the current EDHS algorithm.

Keywords
- Mixed task set.
- Identical multiprocessor.
- Real-time scheduling.
- In-depended tasks.
- EDHS algorithm
- Weakly hard systems

1. Introduction
With the rapid improvement of computer technology, real-time system has been an essential part of human production and life. Real-time system means the correctness of computation highly depends on not only accurate of logic result but also the generation time of the result. As the advanced growth of real-time system, it is increasingly common that
multiple types of real-time tasks exist in the system so that the complexity of real-time system is regularly increased. Therefore the scheduling of the mixed real-time task set (periodic, sporadic and aperiodic) has become a hot research problem, it represents the future direction of real-time systems. Scheduling algorithm is the hard issue of real-time task scheduling, and there are strong theoretical and practical values in the aspect of studying mixed real-time task scheduling algorithm.[1]

Many real-time applications include combined scheduling of hard and soft real-time tasks. Hard real-time tasks have critical deadlines that are to be met in all working scenarios to avoid catastrophic consequences. In contrast, soft real-time tasks (e.g., multimedia tasks) are those whose deadlines are less critical such that missing the deadlines rarely has minimal effect on the performance of the system.[2]

A weakly hard real-time system is a system in which the distribution of its met and missed deadlines during a window of time is bounded. [3] For example, a requirement less than 10 percent of deadlines cannot be missed (hard tasks) represents average information over large period of time and it may mean that 90 percent of deadlines can be missed (soft tasks).

Since the EDHS (Earliest Deadline and Highest-priority Split) scheduling algorithm is needed to be improved to meet evolving of critical real-time system[4,5], our work makes it affective in a weakly hard real-time system. This study focus on mixed task set which consists of 90% periodic and sporadic soft real-time tasks with 10% aperiodic hard real-time tasks in identical multiprocessor environment.

2. Previous studies

Kato & al. [4] came up with EDHS which is a semi-partitioning scheduling algorithm and they chose the success ratio as the key factor to evaluate its performance in the case of first-fit, best-fit and worst-fit on soft sporadic tasks. It has been proved that EDHS improves schedulable utilization by about 10% over P-EDF and this evaluation has been done by using only 16 processors [4].

Authors in [5] have measured the number of tasks migrations, context switches, and average deadline misses and tasks average waiting time of EDHS for soft sporadic tasks by increasing the number of processors in soft real time system.

In [6], a hybrid scheduling approach for real-time systems on homogeneous multi-core architectures have been developed and allows the real-time applications to run with non-real-
time applications concurrently and supports the parallelism between the tasks within an application efficiently.

Some more flexible scheduling algorithms have been proposed in [7]. This work presents several methods to allocate an occurrence of an aperiodic task to a given processor where all its hard real-time requirements have been guaranteed.

There have been some efforts to schedule weakly hard real-time tasks on multiprocessor systems. Wu and Jin proposed the classical weakly hard real-time scheduling algorithms, namely Distance Based Priority (DBP) to apply in multiprocessor applications to guarantee QoS of both hard real-time tasks and multimedia streams even under over load conditions [8]. In fact, the DBP algorithm originally was introduced by [9] on uniprocessor system.

A new fixed-priority scheduling algorithm basing on Rate Monotonic Algorithm (RMA) scheduling policy is presented by Ming and Hai which divides the task into pre-emptive and optional to reflect its urgency basing on weakly hard real-time parameter, and the algorithm can modify the priority of task dynamically during scheduling [10]. It makes the algorithm more flexible. However, the approach is easy to implement and they introduces very low scheduler overhead.

Based on the review, we present an adaptive scheduling algorithm since it can ensure the predictability of system behavior. Hence, the semi partitioned scheduling that EDHS works on is chosen for our schedulability analysis because the advantage of that approach is at any runtime overhead caused by migrations of task can be reduced. Also, the advantages of this approach are its efficiency and simplicity. In most cases, if the task set is fixed and known a priori, the semi partitioning approach is becoming the most appropriate solution.

3. System Model and terms definition
This paper considers the scheduling of n in-depended mixed tasks set with implicit deadlines on a platform of m identical multiprocessors follow:
- 90% of tasks are periodic and sporadic soft tasks.
- 10% of tasks are aperiodic hard tasks.

We consider the task set which contains periodic soft real time task set Tpi which is composed of n periodic soft real-time tasks, sporadic soft real-time task set Tsi which is composed of t sporadic real-time tasks and aperiodic hard real-time taskset Tai which is composed of q aperiodic tasks, which is denoted by TS = \{Tp,Ts, Ta\}.
A periodic task [11,12], in real-time systems, is a task that is periodically released at a constant rate. Periodic soft real-time tasks in our study are $T_p = \{T_{p1}, T_{p2}, \ldots, T_{pn}\}$. Any task $T_{pi}$ of a periodic task set is denoted by the arrival time $a_{pi}$, the deadline $D_{pi}$, the period $T_{pi}$ and the computation time $C_{pij}$ (defined by computing time of task $T_{pi}$ on the resource $R_j$).

The period of sporadic task [11,12] is a minimum inter-arrival time, that is, a minimum interval of time between two successive activations, because a sporadic task is activated irregularly with this bounded rate. Sporadic soft real-time tasks in our study are: $T_s = \{T_{s1}, T_{s2}, \ldots, T_{st}\}$. Any task $T_{si}$ of sporadic real-time task set, the arrival time $a_{si}$, the deadline $D_{si}$, the minimum arrival time $T_{si}$ and the computation time $C_{sij}$ (defined by computing time of task $T_{si}$ on a resource $R_j$).

Ten percent of tasks in our study are aperiodic hard tasks [11,12] that activated irregularly at some unknown and possibly unbounded rate. The aperiodic hard real-time tasks $T_a = \{T_{a1}, T_{a2}, \ldots, T_{at}\}$. Any task $T_{ai}$ of aperiodic soft real-time task set, the arrival time $a_{ai}$, deadline $D_{ai}$, the minimum arrival interval time $T_{ai}$ and the computation time $C_{aij}$ (defined by computing time of task $T_{ai}$ on a resource $R_j$).

One more important parameter that is used to describe a task $T_i$ is its utilization and is denoted as [11,12]:

$$ u_i = \frac{e_i}{p_i} \quad (1) $$

The utilization of a task [13] is the portion of time that it needs to execute after it has been released and before it reached its deadline. $U_{sum}$ denotes the total utilization of a given task set $T$ where as $U_{max}$ describes its maximum utilization.

$$ U_{sum} = \sum u_i \quad (2) $$

A task set $T$ is said to be schedulable on $m$ identical multiprocessor [13] if and only if:

$$ U_{sum}(T) \leq m \quad \& \quad U_{max}(T) \leq 1 \quad (3) $$

4. The proposed algorithm

Our proposed algorithm is inspired from EDHS which is based on the concept of semi partitioned scheduling. However; the difference in our approach compared to EDHS is straightforward:

- Hard aperiodic tasks are first assigned (each hard task assigned to each processor).
- If the migrating tasks arrived at the same time with hard aperiodic task then this condition will be tested:
If (Ci for migrating task + Ci for hard task) \(\leq\) Di for hard task then the migrating task will be executed first, else the hard task will execute first.

Whereas Ci is the computation time for task and Di is the deadline.

- Hard tasks and migrating tasks are not preempt-able.
- Every job of a migrating task begins on the first processor to which it is assigned, and it is sequentially migrated onto the next processor when the assigned capacity is consumed on each processor.
- Partitioned soft tasks are scheduled by LLF.
- If two or more soft tasks have the same laxity then the higher priority will be to the task that has maximum utilization on the processor.

This algorithm we have proposed named LLHS for Least Laxity and Highest-priority Split, since LLF scheduling algorithm is more dynamic than EDF due to its ability to give the highest priority to the task with the least laxity.

The schedulable condition [4] for EDHS to calculate the maximum execution amount \(c's\) that a job in a shared task \(T_s\) can consume on each processor without timing violations when \(di \geq Fps + c's\), the condition is given as follows:

\[
c'_s \leq \frac{di}{F + 1} \left(1 - \sum_{i \in I} \frac{c_k}{p_k}\right) \quad (4)
\]

Otherwise, the condition is obtained as follows [4].

\[
c'_s \leq p_s - \frac{di}{F} \sum_{i \in I} \frac{c_k}{p_k} \quad (5)
\]

Whereas \(di = d - t\) because the scheduling algorithm is EDF, but when it has been replaced with LLF the conditions for a new scheduling algorithm LLHS to calculate the maximum execution amount \(c's\), when \(di \geq Fps + c's\), the condition is given as follows:

\[
c'_s \leq \frac{d_i - e'i}{F + 1} \left(1 - \sum_{i \in I} \frac{c_k}{p_k}\right) \quad (6)
\]

Otherwise, the condition is obtained as follows:

\[
c'_s \leq p_s \frac{d_i - e'i}{F} \sum_{i \in I} \frac{c_k}{p_k} \quad (7)
\]

Whereas \(di - e'i = Li\) is the laxity for the job with implicit deadline.
5. Results and Discussion

In the experiments, the values of the parameters are considered as below, unless mentioned otherwise.

1. Partitioned tasks are preemt-able otherwise, hard tasks and migrating tasks are not preemt-able.
2. The period of periodic and sporadic tasks is a random number with a uniform distribution between 1 and 100.
3. The arrivals for sporadic and aperiodic tasks are a random number between 1 and 100.
4. The number of tasks is 8, 16, 32, 64 and 128 which are executed on 4, 8, 16, 32 and 64 processors respectively, i.e., the number of tasks is double the number of processors (n=2m).
5. We have generated 1000 task sets with full utilization Usum (T) = m.
6. The results have been obtained in an observation window between 1 and 10000.

![Preemption Counts](chart.png)

Figure 1
Comparison of preemption counts per job between LLHS and EDHS

The average number of preemption counts of LLHS and EDHS is shown in Fig. 1. As the number of processors increases, the number of preemption counts of the algorithms increases. In this case, LLHS algorithm shows a better performance with less preemption counts because the migrating tasks and the hard tasks are not preemt-able.
Figure 2
Comparison of context switches counts per job between LLHS and EDHS
The average number of context switches of LLHS and EDHS is shown in Fig. 2. As the number of processors increases, the number of context switches of the algorithms also increases. In this case, LLHS algorithm shows a better performance with less context switches and that for the same reason, the migrating tasks and the hard tasks are not preempt-able.

Figure 3
Comparison of migration counts per job between LLHS and EDHS
Figure 3 shows that the average number of migrations is identical for both algorithms, and that because if a spare capacity of each individual processor is not enough to accept the full execution of the task, then a task is allowed to be shared between multiple processors for both algorithms.
Comparison of tasks average waiting time between LLHS and EDHS

Fig. 4 shows the results of simulations based on tasks average waiting time. We have carried out a study to compare the tasks average waiting time of LLHS and EDHS algorithms. Since a good scheduling algorithm should minimize the waiting time, the results show that the LLHS algorithm outperform the other algorithm that because the hard tasks have least waiting time and LLF algorithm is more dynamic scheduling algorithm and gives the highest priority to the task with least laxity thus it minimizes the waiting time of tasks in the ready queues.

Comparison of success rate between LLHS and EDHS
The success rate is the percentage of tasks that have been executed before its deadlines, and fig.5 shows the comparison results between LLHS and EDHS. The results show that the EDHS algorithm outperform the other algorithm for all task set, but we measure the soft tasks and the hard tasks that miss their deadlines in Fig.6, and the results show that all the deadlines that LLHS has been missed are soft. Otherwise, the EDHS scheduling algorithm has a higher success rate but the deadline misses that EDHS has been missed are soft and hard.

![DeadLine Misses](image)

**Figure 6**
Comparison of deadline misses per job for soft & hard tasks between LLHS and EDHS

6. Conclusion

In this paper, we presented an adaptive real-time multiprocessor scheduling algorithm. Since the concept of time constrain is of such importance in real-time application systems, the proposed LLHS algorithm has developed the EDHS one to be more affective to schedule some bounded percentage of hard tasks in weakly hard systems. The results show that LLHS minimizes the preemptions and the context switches and tasks average waiting time in queues, but it is still needed to be improved to reduce the deadline misses for soft tasks in real-time systems.

Acknowledgment

We would like to express our gratitude and thankful to Dr. Hassan Alahmad and Dr. Samer Suliman for their help, invaluable discussions, and encouragement.
References


