Tardiness Reduction Technique for Multiprocessor Scheduling based on New Assignment Strategy

Ms. Neha Jain  
Computer Science & Engineering  
Sushila Devi Bansal College of Engineering, Umaria, AB Road, Near Rau, Indore

Mrs. Varsha Zokakar  
Information Technology  
Sushila Devi Bansal College of Engineering, Umaria, AB Road, Near Rau, Indore

Abstract: In the real-time system, scheduling is an important factor which decides a schedule for a set of tasks. The process deadline, processor capacity, start time, processing time, and release time of task are not known in prior, which makes the scheduling process complicated. These parameters are well known to as for optimal scheduling algorithm in real-time system. With the increased processing capacity, there is requirement to assign the tasks to processors in defined manner for their effective utilization. The nature of task may require the higher utilization or lower utilization of processors, and sequence of assignment is also become important to avoid higher tardiness. In this paper, an EDF-Fm based algorithm has been simulated for the multiprocessor scheduling. The tardiness in the processes is the key parameter has been considered for the effective scheduling. Various assignment techniques are used for tardiness reduction such as Lowest Execution First (LEF), Lowest Utilization First (LUF) etc. These all are the techniques, based on the assignment priority of jobs. Here we proposed a new concept High-Low High-Utilization-first (HLHUF) for tardiness reduction. From various experiments based on utilization ‘u’, and result analysis, it is concluding that the approach HLHUF gives minimum tardiness.

Keywords- Multiprocessor System, EDF-fm, HLHUF, LUF, LEF etc.

I. INTRODUCTION

A real-time system is one that must execute a program within a specific time constraint. If the time limit is passed before suitable action taken, then the information obtained from sensors would be missing or out dated, then the concept of deadline comes. This is a common issue among all real-time systems models. Some examples of real-time system are as Process Control System, Air Traffic Control System, etc. In real-time system, System collects information from external sensors, analyses the information, process the information and then perform specific action. The deadline of a task is the unit of time before which the task must complete its execution. Based on the timing parameter we can divide the real-time system in two important categories:

(i) Hard real-time system  (ii) Soft real-time system

In hard real-time system, a little bit of delay in program execution is intolerable and entire system will going to crash. Ex: Rocket launching. But in soft real-time system there is a little bit of relaxation on process execution and the system will work fine for little bit of timing delay in program execution. Ex: Web browsing.

If the platform of real-time system is multiprocessor then either all processors are of the same type is known as homogeneous system or when they differ from one another is known as heterogeneous system [1]. Generally two type of scheduling algorithms are used in real-time system, i.e. static and dynamic. In static scheduling algorithm entire scheduling decision are made at compile time but in dynamic scheduling algorithm scheduling decision are made at runtime. Example Earliest Deadline First (EDF).

Scheduling is the most vital parameter in computer system. Generally scheduling strategy decides which task has to be executed at any given time. Generally two type of scheduling algorithms are used in real-time system.

Static scheduling algorithm: - In static scheduling strategy entire scheduling decision are made at compile time [2].

Dynamic scheduling algorithm: - In dynamic scheduling strategy scheduling decision are made at runtime. [3]

A Comparative Study of Scheduling Algorithms for Real Time Task has been described. [4][5] In these many algorithm has been proposed for static and dynamic scheduling algorithm.

The rest of this paper is organized as follows. In section II related work is described and in sec. III system model is described. Sec. IV describes an algorithm includes task assignment, its execution and tardiness bound for EDF-Fm is elaborated. The experiment evaluation and result analysis for reduction of tardiness is described in sec. V and sec. VI summarizes the work.

II. RELATED WORK

Jeremy P. Erickson and James H. Anderson proposed a tardiness bounds technique for soft real time system by splitting jobs, under global scheduling algorithm. The related policy depends on per task worst-case execution times. By splitting job budgets to create sub jobs with shorter periods and worst-case execution times, such bounds can be reduced to near zero for implicit-deadline sporadic task system. However, doing so could potentially cause more preemption and create problems for synchronization protocols [7].
James H. Anderson, Vasile Bud and Uma Maheswari C. Devi have proposed “A new EDF-based scheme that ensure bounded deadline tardiness”. In this scheme, per-task utilization must be capped, but overall utilization need not be restricted. The required cap is quite liberal. This scheme should enable a wide range of soft real-time application to be scheduled with no constraints on total utilization [7].

EDF (Earliest Deadline First) has been proved to be optimal scheduling algorithm for single processor real-time system. It also performs well for multiprocessor system. Limitation of EDF is that its performance decreases exponentially when system becomes slightly overloaded. ACO (Ant Colony Optimization) based scheduling algorithm performs well in both underloaded and overloaded conditions. But its limitation is that it takes more time for execution compared EDF. In this paper, an adaptive algorithm for multiprocessor real-time system is proposed, which is combination of both of these algorithms. The proposed algorithm along with EDF and ACO based algorithm is simulated for real-time multiprocessor system and the results are obtained. The performance is measured in terms of Success Ratio (SR) and Effective CPU Utilization (ECU).

Execution Time taken by each scheduling algorithm is also measured. From analysis and experiments, it reveals that the proposed algorithm is fast as well as efficient in both underloaded and overloaded conditions for real-time multiprocessor systems [8].

Uma Maheswari C Devi and James H. Anderson have proposed a method for “tardiness bounds under global EDF scheduling on a Multiprocessor”. The derive method bound tardiness under preemptive and non-preemptive global EDF, when the total system utilization is not restricted, except that it not exceed the available processing capacity. The tardiness bounds depends on the total system utilization and per task utilization and execution cost, the lower these values, the lower the bound [9].

Under current analysis, soft real-time tardiness bounds applicable to global earliest-deadline-first scheduling and related policies depend on per-task worst-case execution times. By splitting job budgets to create sub-jobs with shorter periods and worst-case execution times, such bounds can be reduced to near zero for implicit-deadline sporadic task systems. However, doing so could potentially cause more preemption and create problems for synchronization protocols. This paper analyzes this trade off between theory and practice by presenting an overhead-aware schedulability study pertaining to job splitting. In this study, real overhead data from a scheduler implementation in LITMUSRT was factored into schedulability analysis. This study shows that despite practical issues affecting job splitting, it can still yield substantial reductions in tardiness bounds for soft real-time system [10].

A scheduling algorithm decides a schedule for a set of tasks. There are numbers of algorithm for scheduling tasks on a processor. Some of these algorithms are used for scheduling tasks on multiprocessor system either under the partitioning scheme or under the global scheduling scheme. The most common scheduling algorithms are: Earliest Deadline First (EDF) and Least Laxity First (LLF). They are optimal scheduling algorithms for single processor system, but problem arises when algorithms are used for multiprocessor system. In this paper, we have proposed a new algorithm, D_EDF. D_EDF scheduling algorithm overcomes the limitations of dynamic algorithm during overloaded conditions. The proposed algorithm D_EDF, simulated and tested for independent, preemptive, periodic tasks on tightly coupled real-time multiprocessor system under global scheduling. The performance is measured in terms of Success Ratio and Effective CPU Utilization. From experiments and result analysis it concludes that the proposed algorithm is very efficient in both underloaded and overloaded conditions. It performs always better than conventional EDF algorithm. The algorithm proposed in the paper performs quite well during overloaded conditions [1].

III. SYSTEM MODEL

Generally task systems consist of N tasks and m identical processors. Consider the scheduling of a task system T, where 1 ≤ i ≤ n is characterized by period and execution time T_i (e_i, p_i). Here e_i is always less than p_i (e_i < p_i).

Each job of T_i has a requirement of execution time unit and deadline in worst-case. The assumption used is that period is equal to deadline (p_i=D_i). The utilization of any task is denoted by u_i and it is calculated by e_i and p_i, where u_i=e_i/p_i. If u is less than or equal to 1/2 then it is called light task and all tasks are considered as light tasks. The total utilization of a task system is sum of all utilizations and all tasks are considered as pre-emptive tasks. In pre-emptive scheduling, tasks have to forcefully relinquish its CPU [7].

In real-time system tasks are divided into two parts, that is: periodic and sporadic tasks. In many real-time systems N recurrent task is denoted as T= {T_1, T_2,...,T_n}. Here consider each task is sequential and characterized by three parameters:

(i) p_i denotes period time.
(ii) e_i denotes execution time.
(iii) D_i deadline of task.

When, p_i>e_i, e_i>0 and relative deadline D_i>e_i, (Because, Ti is sequential so it may not execute on multiple processors at a time). A recurrent task with these characteristics is referred to as a sporadic task [11]. A periodic task is invoked repeatedly in a system at a regular interval. Periodic tasks are released at fixed rates (periods) [7].

Here EDF algorithm is used, in this algorithm jobs with earlier deadline are assigned first to the processor. When a job is assigned to the processor and its execution does not complete within a specific
deadline, then the concept of tardiness comes. When any job misses its deadline it also makes delay in future job releases. Tardiness is defined as highest delay in each task and our goal is to decrease it. We use new assignment technique called High-Low-High Utilization First (HLHUF), which reduces the tardiness in comparison with other assignment techniques like LEF and LUF.

IV. SIMULATION OF EDF ALGORITHM

In the EDF-Fm algorithm, Fm denotes either task is fixed or migrated. Fixed task means its utilization is on one processor. When utilization of any task is shared between two processors then it is called migrated task [7]. An EDF-Fm algorithm contains two phases: (i) Assignment phase (ii) Execution phase. The assignment phase run as a static and execution phase run as dynamic.

Assignment phase: - It is an off-line procedure. In this phase all tasks are assigned to the processor according to their utilization. Here all tasks are considered as a light weight task and total utilization is always less than or equal to sum of processor capacity. Assume all processors have an equal capacity or all processor’s capacity is considered as one.

For assigning tasks to the processor, always it must satisfy some rules. The rules are:
(i) Each assigned migrated task can be shared on at most two processors.
(ii) Each processor contains only two migrated tasks and any number of fixed tasks.
(iii) The sum of share allocated to the any processor is at most one.

The assignment of any task to any processor is based on the following Assignment algorithm:

(i) First calculate the utilization of each task.
(ii) On the basis of utilization, assign first task to first processor till the processor does not fill completely.
(iii) When we assign task to first processor and its utilization is more than process capacity then assign some portion of task to one processor and some portion to other processor.
Example: - When the task utilization is .20 but processor capacity is only .15 left then .15 portions is assigned to one processor and remaining portion, .05 is assigned to another processor. This type of task is called migrated task because it is shared between two processors.
(iv) This process is continuing until all the processors do not fill and all the tasks assignment was not completed [7].

Example: Here we consider a task set comprise with 10 different tasks: T1(9,20), T2(2,10), T3(1,2), T4(1,10), T5(3,15), T6(7,20), T7(3,10), T8(3,10), T9(1,10), T10(1,2) that is assign to 3 different processors. ref. (figure2).

V. TARDINESS BOUND FOR EDF-FM

Generally fixed tasks do not generate tardiness but due to migrated tasks, tardiness is introduced and this tardiness also affects the fixed task execution; as a result, task performance decreases. Our work is to reduce the maximum amount of tardiness by which a job of a fixed task may miss its deadline on each processor. According to assignment rule, each processor contains only one or maximum two migrated task.

Case I: If a processor contains one migrated task, then maximum tardiness in fixed task is:

\[
\Delta = \frac{e_i(f_i,k+1)}{1-s_i,k} 
\]

Case II: If a processor contains two migrated tasks, then maximum tardiness in fixed task is:

\[
\Delta = \frac{e_i(f_i,k+1) + e_i(f_j,k+1)}{1-s_i,k - s_j,k} 
\]

Here e_i is execution of first migrated task and e_j is execution of second migrated task [7].

VI. RESULT ANALYSIS

In this section experiments conducted using randomly generated task sets. The experiments are evaluated on the various task sets for varying number
of processor M and varying maximum values of per task utilization. For each M and u_max 1000, task sets were generated. For generating task sets first we decide utilization (u_i), u_i is varying from .25 to .50 with step of .05, and total utilization is less than M. For each new random task set T_i, first its p_i was generated as a uniform number in the range of [1,100] then e_i, is randomly in range of [1/u_max , u_max*pi] and the last task is generated by sum of all utilization or total utilization of T exactly equaled M[1]. The generated tasks set are classified by the maximum utilization of any task in a task set, u_max. For bounding the tardiness we use various techniques LUF and LUF. Here, a new technique High-Low-High Utilization First is proposed. In this technique, low utilization task is placed in between two high utilization tasks. The result of the following assignment rule is based on the maximum utilization and mean of maximum tardiness.

The below plotted graph shows that High-Low-High Utilization First approach generates minimum tardiness of the two task-assignment approaches. The graph is plotted between maximum utilization and mean of maximum tardiness. Tardiness is quite low in comparisons of LUF and LUF. For u_max= .25, it is approximately 15 percentage low than LUF and other utilization point it always gives minimum tardiness. According to above two strategies, we observe that tardiness was increased when we increase utilization but in the High-Low-High Utilization First (HLHU) strategy tardiness decreases, when we increase utilization. This strategy gives minimum tardiness (best result) of the two assignment strategy. The below plotted graph suggests that (HLHU) may be a reasonable strategy for such task system. ref. (figure 3)

![Figure 3: Tardiness by max. Utilization on 3 Processor](http://www.ijcttjournal.org)

VII. CONCLUSION

In real-time homogenous multiprocessor system various task assignment strategies is used for reducing tardiness which is based on EDF-f_m algorithm but a new task assignment strategy, HLHU First gives minimum tardiness. It gives slightly high tardiness in some points of utilization, compared to HUF and RANDOM strategy, But it gives overall better result than LUF and LUF strategies.

The restriction is not on over all utilization but per task utilization to be at most one-half of a processor capacity. This restriction is not very hard and these assignment approaches are sufficient for scheduling on a soft real-time application. We have only taken first-step ahead to reduce tardiness on EDF-F_m algorithm on multiprocessors. This approach is generally partitioning based scheme, and not supporting dynamic task system, because in these task sets, task parameters are changed in run-time. We defer addressing these issues to future work.

VIII. REFERENCES

[10] Jeremy P. Erickson, James H. Anderson, and Bryan C. Ward “ Fair Lateness Scheduling: Reducing Maximum Lateness in G-EDF-like Scheduling”. Work supported by NSF grants CNS 1016954, CNS 1115284, and CNS 1239135; ARO grant W911NF-09-1-0535; and AFRL grant FA875011-1-0033