An Adaptive CPU Scheduling for Embedded Operating Systems using Genetic Algorithms

Sachin R. Sakhare
Department of Information Technology,
VIIT, Pune, M.S. India
Email:sakharesachin7@gmail.com

Dr. M.S.Ali
PRMCOEM, Badnera,
Amravati, M.S. India
Email:softalis@gmail.com

ABSTRACT
This paper explains novel CPU scheduling approach for embedded operating Systems. In this approach we have used genetic algorithm (GA). Proposed Adaptive algorithm combines both EDF and GA based algorithms. Basically the new algorithm uses EDF algorithm but when the system becomes overloaded, it will switch to GA based scheduling algorithm. Again, when the overload disappears, the system will switch to EDF algorithm. Therefore the proposed algorithm takes the advantages of both algorithms and overcome their limitations.

We have simulated, proposed adaptive algorithm along with both EDF and GA based algorithms for real time systems. The performance is measured in terms of Success Ratio and Effective CPU Utilization. From analysis and experiments it reveals that the proposed algorithm is fast as well as efficient in both under loaded and overloaded conditions.

Keywords - Real time Scheduling, Genetic Algorithms, Embedded Operating Systems, Earliest deadline first, scheduling algorithm.

1. INTRODUCTION
Overall, it is hard to come up with a comprehensive answer to the question of what is Embedded Operating System (EOS). Perhaps, the simplest answer would be “an operating system designed for embedded systems”. However it often refers to a collection of components or design approach for real-time software systems. The fundamental difference between the traditional general purpose operating systems (GPOS) and EOS is the need for “deterministic” timing behavior in case of real time systems. In addition, EOS must facilitate ways to handle unexpected errors, missed deadlines, inevitable failures, and may operate under resource constraints. Real-time task scheduling is one of the interesting topics in the context of EOS. The interest in the topic started with the seminal work of Liu and Layland in 1973. Since then many algorithms have been proposed for real time scheduling. Schedulability analysis is a fundamental aspect of real-time scheduling. A set of task is said to be schedulable if enough CPU time is available to execute all these tasks before their deadlines. Each real time task is assigned a priority and a deadline.

The real time tasks are of 3 types [1]. If a task needs to be executed after regular time interval then it is called periodic task. If a task’s relative activation time is not known then it is non-periodic task. A non-periodic task with a hard deadline is called sporadic task. Scheduling algorithm in REOS must schedule all periodic and sporadic tasks such that their timing requirements are met.

Task scheduling can be either performed preemptively or non-preemptively and either statically or dynamically. For small applications, task execution times can be estimated prior to execution and the preliminary task schedules statically determined. Two common constraints in scheduling are the resource requirements and the preference of execution of the tasks. Typical parameters associated with tasks are:

- Average execution time
- Worst case execution time
- Dispatch costs
- Arrival time
- Period (for periodic tasks).

The objective of scheduling is to minimize schedule-length, average tardiness or laxity, and to maximize average earliness and number of arrivals that meet deadlines. Real-time scheduling algorithms can be classified into two categories: static priority algorithms and dynamic priority algorithms.

The most commonly used static priority driven preemptive scheduling algorithm for periodic tasks is the Rate Monotonic (RM) scheduling algorithm [2]. The RM algorithm assigns different priorities proportional to the frequency of tasks. If a successful schedule cannot be found using RM, no other fixed priority scheduling system will avail. But the RM algorithm provides no support for dynamically changing task periods and/or priorities.

Earliest deadline first (EDF) scheduling can be used for both static and dynamic real-time scheduling. Its complexity is O(n2), where n is the number of tasks, and the upper bound of process utilization is 100%. A variant of EDF is Minimum Laxity First (MLF) scheduling where a laxity is assigned to each task in the system and minimum laxity tasks are executed first. MLF considers the execution time of a task, which EDF does not. Another variant of EDF is the Maximum Urgency First
MUF) algorithm, where each task is given an explicit description of urgency [3].

Scheduling problems with deadlines which characterize real-time systems are almost always shown to be NP-hard in either single processors or multi-processors. Various heuristic approaches have been widely used for scheduling. The use of genetic algorithm (GA) for real-time task scheduling is now been studied extensively. GAs seem attractive to real-time application designer as it relieves the designer from knowing how to construct a solution and the designer just requires knowing how to assess a given solution[4]. Genetic algorithm has been utilized to minimize the total execution time. The simulation studies presented in this paper shows the efficiency of the GA based adaptive scheduler compared to other schedulers.

The whole paper is organized as follows: In Section.2 the proposed algorithm is explained and discussed. Section 3 contains simulation method and performance measuring parameters. Section 4 contains the results obtained and the paper ends with a brief conclusion in Section.5.

2. PROPOSED ADAPTIVE ALGORITHM

The Adaptive algorithm is combination of two scheduling algorithms: EDF algorithm and GA based Scheduling algorithm.

2.1 EDF Algorithm

The priority of each task is decided based on the value of its deadline. The task with nearest deadline is given highest priority and it is selected for execution. This algorithm is simple and proved to be optimal when the system is preemptive, underloaded and there is only one processor [7].

2.2 GA Based Scheduling Algorithm

The first step of GA is to encode any possible solution of the problem as a set of strings called as chromosomes. The genetic algorithm used in our system is given below.

1. Randomly initialize population (n)
2. Determine fitness of population (n)
3. Repeat
   a. Select parents from population (n)
   b. Perform 2 point partially matched crossover on parents creating population (n+1)
   c. Perform 1 bit mutation of population (n+1)
   d. Determine fitness of population (n+1)
5. Until best individual is good enough.

2.2 Adaptive Scheduling Algorithm

Proposed Adaptive algorithm combines both of these algorithms and it works as per following:

- During underloaded condition, the algorithm uses EDF algorithm i.e. priority of the job will be decided dynamically depending on its deadline.
- During overloaded condition, it uses GA based Scheduling algorithm i.e., priority of the jobs will be decided depending on the fitness value of each chromosome. Fitness value of chromosome is decided by the Success Ratio(SR) which is defined as:

\[ SR = \frac{\text{Number of tasks successfully scheduled}}{\text{Total number of tasks arrived}} \]

Switching Criteria:

- Initially the proposed algorithm uses EDF algorithm considering that the condition is not overloaded. But when a job has missed the deadline, it will be identified as overloaded condition and the algorithm will switch to GA based scheduling algorithm. After 10 jobs have continuously achieved the deadline, again the algorithm will shift to EDF algorithm considering that overloaded condition has been disappeared.
- During under loaded condition, EDF algorithm is used for reducing execution time and during overloaded condition GA based scheduling algorithm is used for achieving better performance. By this way, adaptive algorithm has taken advantage of both algorithms and overcome their limitations.

3. SIMULATION METHOD

We have implemented EDF, GA based & the adaptive algorithms and have run simulations to accumulate empirical data. We have considered periodic tasks for taking the results. For taking result at each load value, we have generated 200 task sets each one containing 3 to 9 tasks. The results for 30 different values of load are taken (0.5 ≤ load ≤ 5) and tested on more than 35,000 tasks. The simulation programs are executed on Intel Core 2 Duo machine with 2 GB SDRAM and Linux Red Hat Operating System. Here, periodic tasks load (L) of the system can be determined by using the following equation:

\[ L = \sum_{i=1}^{n} \frac{E_i}{D_i} \]

where,

- n = Number of tasks
- E = Execution time required by the task
- P = Period of the task
- D = Deadline of the task

The system is said to be overloaded when, value of L ≥1. A reasonable way to measure the performance of a scheduling algorithm during an overload is by the number of task the scheduler can feasibly schedule according to the algorithm. We have considered
following three as our main performance measuring criteria:
Success Ratio (SR) is one of the appropriate performances metric, as meeting deadline is important in real time task scheduling and it is defined as:

$$SR = \frac{\text{Number of tasks successfully scheduled}}{\text{Total number of tasks arrived}}$$

Effective CPU Utilization (ECU) gives information about how efficiently the processor is used and it is defined as,

$$ECU = \frac{\sum_{i \in S} V_i}{T}$$

Where,
- $V$ is value of a job and,
  - Value of a job = Execution time of a job, if the job completes within its deadline.
  - Value of a job = 0, if the job fails to meet the deadline.

3) The execution time required by each scheduling algorithm is very important especially when we are working with real-time systems [10, 11].

The results are obtained, measured in terms of SR & ECU, compared with EDF algorithm and GA based scheduling algorithm in the same environment and shown in following section.

4. RESULTS

Following figure.1 & figure.2 represents the results obtained from simulation study. Fig. 1 shows the results obtained in terms of %SR and %ECU during overloaded conditions, using the same algorithms. Fig. 2 shows the comparison of the execution time taken by each algorithm.

![Figure 1: Load Vs %SR](image1)

![Figure 2: Load Vs %ECU](image2)
Figure 1 shows the results obtained in terms of %SR Vs Load & Figure 2 shows the results obtained in terms of %ECU Vs Load. From the results we can observe that the adaptive algorithm performs better than EDF algorithm during overloaded conditions even GA based algorithm performs better than EDF algorithm during overloaded conditions.

5. CONCLUSIONS

In this paper, we have proposed GA based adaptive scheduling algorithm for scheduling periodic tasks on single processor environment when the tasks are preemptive. The results achieved during simulation prove the following:

• The proposed adaptive algorithm is more efficient than EDF for single processor, preemptive environment when the system is overloaded.

• EDF algorithm does not perform well when the system is overloaded and GA based scheduling algorithm takes more execution time in that type of condition. These are the main limitations of both algorithms.

• During underloaded condition, the execution time taken by the proposed algorithm is almost same as EDF algorithm (i.e. less time).

• The algorithm can switch automatically between EDF algorithm and GA based scheduling algorithm.

Therefore, the proposed adaptive algorithm is very useful when future workload of the system is unpredictable.

REFERENCES


