www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 8 Aug 2015, Page No. 13838-13841

Aperiodic task Scheduling Algorithms for Multiprocessor systems in Real Time environment

Nirmala H[1], Dr.Girijamma [2]
[1] Research Student, [2] Professor, Dept of CSE, RNS Institute of Technology, Bangalore

Abstract
Multiprocessor systems contains multiple processors either homogeneous or heterogeneous, scheduling tasks for such system is very critical and hence scheduling protocol should be followed for optimality.
Scheduling algorithms gives the scheduler a set of protocols to manage the real time systems. In this paper we present an overview of aperiodic task scheduling algorithms servers for real-time systems on multiprocessor systems and method is proposed for the aperiodic task having communication delay which can be scheduled using a Genetic Algorithm (GA)
.

Keywords: Aperiodic task, Genetic algorithm, DAG

1. Introduction

In Real-Time computing the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced [1]. These systems are used in many ways today more than PCs, we don't know or think about it when we use the devices in which they are embedded, Cars, planes and entertainment systems are some devices in which real-time systems reside [2].
In a real-time application, tasks are the basic executable entities that are scheduled. The tasks may be periodic or aperiodic and may have soft or hard real-time constraints. Scheduling a task set consists of planning the order of execution of task requests so that the timing constraints are met. Scheduling aperiodic tasks with different WCETs of task at different criticality levels is very difficult topic to workout [3].
Real-time task scheduling could be done either statically or dynamically. Dynamic scheduling for a set of tasks is computed at run-time based on the tasks that is really executing. Static schedule on the other hand is done at compile time for all possible tasks. [4]. Example for Static algorithms is Rate monotonic (RM) scheduling algorithm is a uniprocessor static-priority preemptive scheme. Dynamic schedule for a set of tasks is computed at run-time based on the tasks that is really executing. Earliest deadline first (EDF) is a dynamic priority driven scheduling algorithm which gives tasks priority based on deadline.
Rest of the paper is organized as follows scheduling is discussed in section 2, overview of aperiodic scheduling algorithms in section 3, proposed scheduling approach in section 4 followed by conclusion.

2. Background

Two main goals of task scheduling in real-time systems are: meeting deadlines and achieving high resource utilization. The two main approaches of scheduling algorithms on multiprocessor are global scheduling and partitioned scheduling. Global scheduling choose a task and assign it to one of the idle processors otherwise, preempt one of the running task. It is an on-line processor assignment with migration. They are well suited for multiprocessor architectures. A preemptive scheduler suspends the execution of currently running task when a higher priority task enters. Dynamic scheduling is an online scheduling of new tasks, the scheduler dynamically determines the feasibility of scheduling them without jeopardizing the guarantees that have been provided for the previously scheduled tasks. Online scheduling is a server based scheduling hence we need to look up on different severs available to sever aperiodic tasks. Few of them are discussed here.

Partitioning scheduling choose a processor for tasks, and then run local scheduler on each processor, there is no migration and may apply end-to-end worst case response time analysis. It is an offline assignment.

2.1. Serving aperiodic tasks

Many of the systems need algorithms for scheduling of both periodic and aperiodic tasks. Tasks which arrive at unpredictable time are aperiodic and it is difficult to guarantee a response time. Aperiodic task set is allowed to execute within the server task. Deadline of an aperiodic task is not used in scheduling decision instead deadline of sever is used. Bandwidth of the server and execution time of the server is used. The sever algorithms presented improve the average response time.

The Background server algorithm [5] is a simple technique for serving soft deadline aperiodic tasks, here aperiodic tasks are served in the background and periodic tasks get priority over aperiodic tasks hence results in bad response time for them.

The polling server [5] is a periodic server here aperiodic tasks are served at the beginning of each of the polling server's

periods , if there are no request pending the server suspends itself allowing periodic tasks to execute. The server is treated as a hard deadline periodic task with a fixed execution time budget, whose deadline is equal to its period. i.e Ti=Di. Where Ti is period and di is deadline of a task .If we have multiple servers at different priority levels can accommodate a set of tasks with a range of hard and soft deadline request.

The deferrable server (DS) algorithm [6] is an algorithm which focus on quick response times for aperiodic tasks. This algorithms improves the average response time for aperiodic tasks compared to background and polling server. The DS is periodic task with period and capacity, which serves aperiodic tasks and continue to do so until available capacity runs out, or the end of the period is reached. Regular periodic tasks are ready at set times whereas the DS may receive requests at any time during the period and therefore executes at different times. The algorithm preserves the execution time allocated for aperiodic service if, upon the initiation of the server task, no aperiodic request are pending. It maintains the aperiodic server's execution time budget for the current period, as long as it has not been exhausted. The DS uses fixed periodic scheduling method (Rate Monotonic scheduling). The server has the highest priority preferable if we want good responsiveness and ensure that aperiodic tasks meet their deadlines, while assigning the DS medium priority aperiodic tasks might miss their deadlines as other tasks might preempt the server.

The deadline deferrable server algorithm (DDS) [6] is able to serve requests that come during the middle of the period. The DDS scheduler serves aperiodic requests at a priority consistent independent of the actual arrival time. They served with the request having come in at the beginning of the period, so long as the server's execution time budget has not exhausted.

The priority Exchange server (PE) [5] is another capacity preserving scheduling algorithm that uses a periodic server. The algorithm mimic the DS algorithm but differs in how capacity is preserved. The server replenishes its capacity at the beginning of each period. Aperiodic tasks waiting to be served at the start of a new period will be executed at the priority of the server and consume capacity, assuming the server currently has the highest priority. On the other hand, if there are no aperiodic tasks ready for execution the server allows a ready lower-priority periodic task to execute in exchange for accumulation of capacity at the priority level of that periodic task. Whenever an aperiodic task requests to run capacity available for the server at the task with the highest priority, amongst those with which capacity has been exchanged, will be consumed by the aperiodic task. This type of capacity exchange continues on lower levels with periodic tasks, hence server capacity will not be lost, just stored at lower priority levels or consumed by aperiodic requests unless at some point the capacity is exchanged with the idle task.

The DS algorithm is not as complex as the PE algorithm due to the way capacity is preserved at the priority of the server. In order for both algorithms to function properly particular resource utilization has to be reserved for the server. The server utilization, U_S, is the ratio of execution time to the period, directly affects the schedulability of the system. The highest utilization bound for periodic tasks at which the periodic tasks can be scheduled, U_P, is determined by RM.

$$DS : U_P = ln\frac{U_S + 2}{2U_S + 1}$$

$$PE : U_P = ln\frac{2}{U_S + 1}$$

From the equations, for any given value, U_S, where $0 < U_S < 1$, the schedulability bound, U_P, is lower for the DS algorithm than for the PE algorithm. Another indication is that for a given Up the server utilization is lower for the DS algorithm than for the PE algorithm.

The sporadic server (SS) algorithm preserves the unused high priority execution time indefinitely. This is an improvement over DS. The schedulabling effect of tasks with lower priorities cannot be worse than that of a periodic task with same period and execution time equal to the server size.

The sporadic server (SS) [5], similar to the DS, consists of a periodic server for aperiodic tasks but how it replenishes capacity differs. The server checks at which time in the future capacity will be replenished depending on when aperiodic requests occur and the priority of the current executing periodic tasks. The explanation of the algorithm involves the following terms:

_ P_{exe} the priority level of the currently executing task in the system.

_ P_S the priority level of the sporadic server.
_ Active is used to describe the priority level when $P_{exe} > P_S$.
_ Idle is the opposite of active, $P_{exe} < P_S$.
_ RT_S is the time at which the server replenishes consumed capacity.

The sporadic server starts [5] with fully replenished capacity. Whenever the server becomes active RT_S is set to the current time added to the period of the server. How much the server should replenish at RT_S is determined when the server becomes idle or all the capacity has been consumed. The amount to replenish is the capacity consumed from the point the server was activated to the point it becomes idle or runs out of capacity. The sporadic server performance is better than the background server and the polling server. The performance is comparable to the DS and PE algorithms although they are in some cases inferior to the SS algorithm.

The deadline Sporadic sever (DSS) algorithm assign priority to the task through an appropriate choice of deadline .It aim to budget the server's execution time in such a way that the effect of the server on the schedulability of hard-deadline tasks is no worse than that of a hard-deadline periodic task

DOI: 10.18535/ijecs/v4i8.40

with period Ts & deadline and execution time Cs. Here sever execution time budget is assigned in chunks.

The algorithm that we chose will greatly influence the behavior of a real-time system and for this reason there are many algorithms available and still research is going on in this direction.

3. Proposed method

AS we are interested in aperiodic task which has a deadline by which it must finish or start or it may have a constraint on both start and finish time. Response of aperiodic tasks may be prohibitively long and there is no possibility to assign a high priority to them. When new aperiodic task come we need to schedule it according to either start time or finish time as shown it the figure 1.

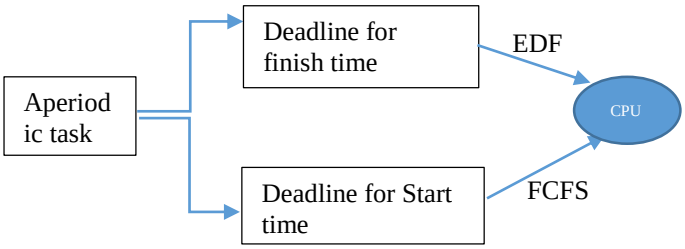

Figure. 1 Allocation of aperiodic task for different servers

3.1 Algorithm

Aperiodic task with execution time, deadline for start time and finish time and communication delays if exists.

1. If aperiodic tasks have equal inter arrival time, they will be assigned to sporadic server.
2. Otherwise check for communication delay exist in the task control block of the task then go to step 4 , if no assign it to Background server.
3. Check with which task it communicate assign it to the server where that task is executing or assign it to new processor.
4. If communication delay is found in task control block we can construct DAG and apply genetic algorithm for scheduling.

Genetic algorithm is an approach for finding approximate solution for optimization. It has initial population, fitness operator function, selection operator, crossover operator and mutation operator. A GA [9] starts with a generation of individual, which are encoded as strings known as chromosome. A chromosome corresponds to a solution to the problem. A fitness function is used to evaluate the fitness of each individual. In general, GAs consists of selection, crossover and mutation operations based on some key parameters such as fitness function, crossover probability and mutation probability.

3.2 Direct acyclic graph (DAG)

A directed acyclic graph [8] is represented as G=(T,E) where T is a set of nodes which represents the tasks, and E is the set of edges which represent the execution dependencies as well as the communication cost between two tasks on different processor. Suppose T consists of m non preemptive tasks as: $T= \{t_j : j=1, 2, 3,4, ..., m\}$. A directed edge set E consists of k edges ranging from k=1, 2, …, r . Suppose any two task t 1 and t2 € T having a directed edge e1 i .e. an edge from t1 to t2 which mean that t2 cannot schedule until t1 has been completed, t1 is predecessor of t2, t2 is the successor of t1, under the relation of dependency on multiprocessor system as shown in figure 2.

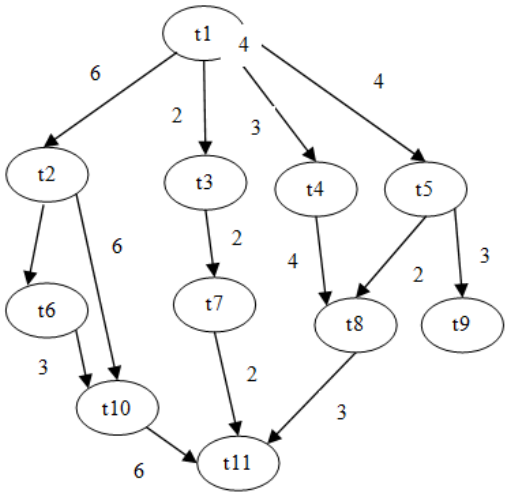

Figure 2. A DAG for task set

4. Conclusion

The paper gives a brief insight about different servers for aperiodic task scheduling. The proposed algorithm using Genetic algorithm would optimizes the response time that will be implemented and results will be discussed in the future.

References

[1] William Stallings “Operating Systems Internals and Design Principles”, sixth Edition, Pearson Prentice Hall.

[2]Fredrik Lindh, Thomas Otnes, Jessica Wennerström “Scheduling Algorithms for Real-Time Systems”, Mälardalens niversity, Sweden.

[3] Risat Mahmud Pathan, “Schedulability Analysis of Mixed-Criticality Systems on Multiprocessors,” *24th Euromicro Conference on Real-Time Systems (ECRTS), Pisa, Italy, 2012*

[4] Betzy Varghes, Alamgir Hossain, and Keshav Dahal “Scheduling of Tasks in Multiprocessor System Using Hybrid Genetic Algorithms”, Springer-Verlag berlin Heidelberg 2009.

[5] Sprunt B., "Aperiodic Task Scheduling for Real-Time Systems" Ph.D. Dissertation, Department of Electrical and

Computer Engineering, Carnegie Mellon University, August 1990.

[6] Strosnider J. K., Lehoczky J. P. and Sha L., "The Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments", IEEE Transactions on Computers, vol. 44, no. 1, January 1995.

[7] Annam Swetha, Radhamani Pillay,Sasikumar Punnekkat and Santanu Dasgupta, " Hard Aperiodic scheduling in Fault tolerant Cruise system- Comparative Evaluation with Dual Redundancy", Springer International Publishing Switzerland 2015.

[8] Sri Raj Pradhan, Sital Sharma, Debanjan Konar, "A comparative study on Dynamic scheduling of Real-Time tasks in Multiprocessor system using Genetic algorithms", IJCA (0975-8887), volume 120-No 20, June 2015.

[9] Jasbir Singh, Gurvinder Singh," Improved Task Scheduling on Parallel System using Genetic Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 39– No.17, February 2012