Implementation of Adaptive Scheduling Algorithm Using Hierarchical Real-Time Scheduling Framework

Swapnil.S.Nehar S.S.Sherekar Dr. V. M. Thakare

Abstract **:- Scheduling algorithm is a hot research topic of real time system. The quality of real-time scheduling algorithm has a direct impact on real-time system's throughput capacity, response time, even on tasks' scheduling result in required deadline. In this work; study of the some technologies i.e. EDF, ACO, Adaptive Process Scheduling, Priority-Based scheduling and Hierarchical Real-Time Scheduling Framework is carried out and it is used for scheduling of process in real time environment. These methods have been analysed with their accuracy, efficiency, success ratio of scheduling but has some limitations. So a new framework by combining above methods can be proposed. The proposed method provides better success ratio in scheduling real time processes to other existing methods.**

Keywords **: Real-time system, scheduling algorithms, EDF, Adaptive Process Scheduling.**

I. INTRODUCTION

In real time environment, scheduling the processes to achieve their deadline is the main constraint for improving the system performance. Timing constraints demand the related tasks are completed in prescribed time limit. This property of real time system largely depends on real-time tasks' scheduling algorithms. The quality of real-time scheduling algorithms has a direct impact on real-time system's throughput capacity (tasks number processed in system unit time), response time, even on tasks' scheduling result. An adaptive Scheduling algorithm dynamically adapts by making use of EDF and ACO algorithm. To manage under loaded and overloaded condition adaptive scheduling algorithm switched from EDF to ACO algorithm [1]. Small improvement

in disk speed considerably improves the system performance. OCBP-schedulability that offers a performance guarantee (as measured by the processor
speedup factor) that is superior to the performance guarantee offered by the worst case reservations schedulable (WCR) approach. OCBP-schedulability is a constructive test: It can be determine offline, before knowing the actual execution times, a total ordering of the jobs in a priority list and for each scenario execute at each moment in time the available job with the highest priority. *Hierarchical Real-Time Scheduling Framework* (HRTSF) has been investigated to schedule real-time tasks to meet their deadlines by sharing resources hierarchically under various scheduling algorithms. In the hierarchical real-time scheduling framework, a system is usually composed of multiple levels of components, where independent components in a level share a computational resource allocated from a parent component to its child components.

II. BACKGROUND

There are many kinds of real-time system that have different resources demand and different forms of timing constraints. The corresponding scheduling algorithms are different. It makes us difficult to compare the scheduling algorithms applied in different real-time systems. For example, if the task is periodic, its period is very important. If the task is aperiodic, focus is on its deadline. Periodic task may have a deadline. The task must be finished before it. Deadline may be equal or unequal the period. Periodic and aperiodic task may both have starting timing constraints. In static situation, off-line scheduling seeks for all tasks' deadline to be met. If the scheduling strategy is not the only one, the one whose average start time is larger is chosen. In a dynamic real-time system, it cannot assure all deadline be met in advance. So the strategy that meet more deadline is chosen.[1]

Hierarchical Real-Time Scheduling Framework (HRTSF) has been investigated to schedule real-time tasks to meet their deadlines by sharing resources hierarchically under various scheduling algorithms. A Hierarchical Real-Time Scheduling Framework is a scheduling framework to support real-time systems which are composed of multiple levels of real-time components with a hierarchy. Multiple components in each level share a computational resource allocated from their parent component[2]. OCBP-schedulability is a constructive test: it determine offline, before knowing the actual execution times, a total ordering of the jobs in a priority list and for each scenario execute at each moment in time the available job with the highest priority[3]. An Adaptive algorithm combines the EDF and ACO algorithm. Table 2[4] shows the execution of process using an Adaptive algorithm. Now all the process entered into an Adaptive queue. All the entered processes are in waiting state and depend upon the release time the process status change to the running. Load of each process is calculated if the calculated load is greater than the given CPU load then the process is switch to ACO algorithm [4].

III. PREVIOUS WORK DONE

The classification of real-time scheduling has a variety of ways. According to the scheduling table and schedulability analysis (off-line or on-line), real-time scheduling can be divided into static scheduling and dynamic scheduling. According to system environment, there are uniprocessor scheduling, centralized multi-processor scheduling and distributed scheduling. According to whether the task could be preempted or not, it can be divided into preemptive scheduling and non-preemptive scheduling. There are hard real-time

scheduling and soft real-time scheduling, according to real-time requirement. The scheduling algorithms also categorized on the basis of uniprocessor scheduling, centralized multi-processor scheduling and distributed scheduling.

a) RMS

The scheduling algorithm in which task is assigned by rate monotonic priority (RMPA) is called Rate Monotonic Scheduling (RMS). In RMPA, tasks' priorities are assigned by their periods Ti. The task whose period is shorter has a higher priority. The longer period task has a lower priority. The task whose period is the shortest is run firstly.

b) EDF

Earliest-Deadline-First (EDF) is also called Deadline-Driven- Scheduling (DDS) [1]. In EDF, the periodic task which has the earliest deadline will be assigned the highest priority. The task which has the highest priority, namely the task which has the earliest deadline is scheduled firstly. Different from RMS, tasks' priorities are changed over time in EDF. At each schedulable moment, tasks' priorities will be changed dynamically according to their earliest deadlines. [1]

As real-time systems become complex and require high-performance, Hierarchical Real-Time Scheduling Framework (HRTSF) has been investigated to schedule real-time tasks to meet their deadlines by sharing resources hierarchically under various scheduling algorithms. [2]

OCBP-schedulability is a constructive test: it determine offline, before knowing the actual execution times, a total ordering of the jobs in a priority list and for each scenario execute at each moment in time the available job with the highest priority. The priority list is constructed recursively using the approach commonly referred to in the real time scheduling literature as the "Audsley approach"; it is also related to a technique introduced by Lawler. First determine the lowest priority job: Job Ji may be assigned the lowest priority if there is at least Pi (Xi) time between its release time and its deadline available when every other job Jj is executed before Ji for Pj (Xi) time units. [3].

IV. EXISTING METHODOLOGIES

a) Hierarchical Real-Time Scheduling Framework:

A Hierarchical Real-Time Scheduling Framework (HRTSF)[2] is a scheduling framework to support real-time systems which are composed of multiple levels of real-time components with a hierarchy. Multiple components in each level share a computational resource allocated from their parent component. Thus, the basic unit of the HRTSF is a component which is defined by (*W, R, A*) as follows.

- *W*: workload or a set of tasks in a component
- *R*: resource model supported from a upper level
- *A*: scheduling algorithm of *W* for a given resource model *R*

A workload set *W* is composed of periodic real-time tasks on a single processor. Each task τi is defined by (*pi, ei*), so that a task τi releases a job of *ei* execution time per every *pi* time units. It is also assume that the deadline of τi equals to *pi*.

Load	Process	Thread	Handles	CPU Usage	Memory Used MB
10	40	908	22315	2.34	35.09
15	42	879	21619	3.00	39.33
20	47	820	21559	4.56	40.11
25	51	789	21499	5.89	43.33
30	57	756	21434	6.99	44.55
35	62	715	20994	10.77	46.33
40	66	700	21222	15.33	48.33
45	70	689	21455	18.32	50.00

TABLE I. CPU USAGE AND TOTAL MEMORY CONSUMPTION

A resource model *R* specifies the amount of resource allocated from a parent or upper-layer component. It use the periodic resource model $\Gamma(\Pi, \Theta)$ where Π is the resource supply period and Θ is the resource supply time per period.

Process ID	Process Name	Start Time	Dead line	Execution Time	Release timein Sec	State	Priority	CPU Load
1237	P1	10:14:26	10:18:26	4	1	Run	Normal	9.19
1238	P2	10:15:10	10:18:60	3	2	Wait	Normal	5.68
1239	P3	10:16:57	10:19:19	3	1	Run	Normal	11.22
1240	P4	10:17:44	10:18:55	1	1	Run	Normal	14.55
1241	P5	10:18:23	10:20:45	2	1	Run	Normal	14.55
1242	P6	10:19:56	10:24:34	5	1	Run	Normal	7.89
1243	P7	10:20:34	10:22:12	2	4	Wait	Normal	11.22
1244	P8	10:21:22	10:22:23	1	3	Wait	Normal	12.34
1245	P9	10:22:46	10:24:33	1	2	Waiting	Normal	13.34
1246	P10	10:23:06	10:25:29	2	2	Waiting	Normal	10.00

TABLE II. ADAPTIVE SCHEDULING ALGORITHM QUEQUE

A component with the resource model $\Gamma(\Pi, \Theta)$ is guaranteed to be supplied with Θ resource time every Π time units from its upper-layer component. In contrast to the resource model, the interface model takes the role of abstracting a component with collective real-time resource requirements In the HRTSF, the interface model *I* is defined by (*P, E*), where *P* is period and *E* is resource supply time. If the resource with the amount of *E* is supported per the period *P* from upper level, then the feasibility condition for a component is guaranteed. For example, Figure 1[2] shows an example of the hierarchical real-time scheduling framework. The component *C*1 has two periodic real-time tasks and scheduled by Earliest Deadline First (EDF). The interface *I*1 is the real-time component interface by merging those two requirements as a single real time requirement. Similarly, *C*2 contains two tasks scheduled by Rate Monotonic (RM) and provides the interface model *I*2. To the higher level component *C*0, the two component abstraction models, *I*1 and *I*2 are considered as two real-time periodic tasks. Therefore, the component considers two subcomponents as sub-tasks with periodic real-time requirements based on their interface models, which makes it easier to compose multi-layer real-time systems. [2].

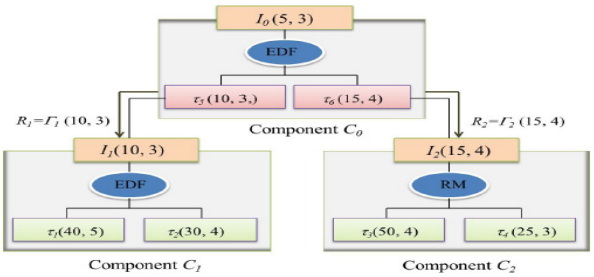

Fig. 1. An example of HRTSF

b) PRIORITY-BASED SCHEDULING:

OCBP-schedulability, that offers a performance guarantee (as measured by the processor speedup factor) that is superior to the performance guarantee offered by the worst case reservations schedulable (WCR)[3]-approach. OCBP-schedulability is a constructive test: it determine offline, before knowing the actual execution times, a total ordering of the jobs in a priority list and for each scenario execute at each moment in time the available job with the highest priority. The priority list is constructed recursively using the approach commonly referred to in the real time scheduling literature as the "Audsley approach"; it is also related to a technique introduced by Lawler . First determine the lowest priority job: Job Ji may be assigned the lowest priority if there is at least Pi (Xi) time between its release time and its deadline available when every other job Jj is executed before Ji for Pj (Xi) time units (the WCET of job Jj according to the criticality level of job i). This can be determined by simulating the behavior of the schedule under the assumption that every job other than Ji has priority over Ji .The procedure is repeatedly applied to the set of jobs excluding the lowest priority job, until all jobs are ordered, or at some iteration a lowest priority job does not exist. [3].

c) PATTERN BASED DISK SCHEDULING ALGORITHM,

Pattern based disk scheduling algorithm is combination of an adaptive scheduling algorithm used for managing real time processes, an HTBS algorithm for predicting the disk block requests and CAP pattern detecting algorithm for predicting buffer cache block access patterns. Modules of pattern based real time disk scheduling algorithm for virtualized environment is shown in "Fig. 2." [4]

B. Non-Working Conserving Disk Scheduling Module

Real time processes issue request to blocks of disk that are shared in the virtual environment. To reduce disk access latency HTBS is used by disk scheduler module. HTTS dispatches the disk requests by maintaining spatial locality, which improves the overall system performance in terms of seek time. [4]

C. CAP Pattern Detection

Buffer cache pattern detector module uses CAP pattern detection algorithm to identify the patterns of the buffer cache blocks and predict their future access request. CAP makes use of program counter and identifies once-identified, frequently identified, recently identified and unidentified patterns in the accessed blocks. [4].

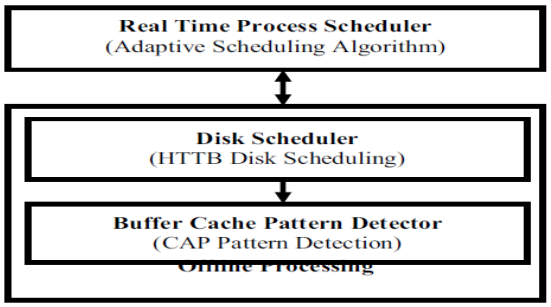

Fig. 2: Modules of Pattern Based Detection Algorithm

V. ANALYSIS AND DISCUSSION

Real-time system that have different resources demand and different forms of timing constraints. The corresponding scheduling algorithms are different. It makes difficult to compare the scheduling algorithms applied in different real-time systems. The scheduling strategy has an obvious impact on predictability which means that how to predict the task whether meet the deadline demand or not more accurately when a specific scheduling method is applied in real-time system. real-time scheduling can be divided into static scheduling and dynamic scheduling. According to system environment, there are uniprocessor scheduling, centralized multi-processor scheduling and distributed scheduling.

Uniprocessor real-time scheduling:

Suppose task set $S=\{\tau_1,\tau_2,\cdots,\tau_n\}$their periods are $T_1,T_2,\cdots,T_n$runtime is $c_1,c_2,\cdots,c_n$. Deadline is $D_1,D_2,\cdots,D_n$., $D_i=T_i$. Task τ_i can be preempted. The utilization ratio of CPU is $U=\sum_{i=1}^{n}(c_i/T_i)$. For uniprocessor scheduling, if $U\leq 1$, S is schedulable.

Multi-processor real-time scheduling:

At present, the research priority of real-time system has turned from uniprocessor scheduling to multi-processor scheduling. But the problem of uniprocessor scheduling and synchronization still need to be concerned in detail when real-time system is verified. The reason is that every task is assigned to a subsystem in static priority-driven multiprocessor system. Every subsystem owns a processor. The task is scheduled by the processor it is on. Scheduling algorithms of multiprocessor are more complicated than that of uniprocessor. In order to meet deadline, the system should have powerful processing capabilities. At the same time, hard real-time system needs high reliability and ability to process redundant information. The advantage of multiprocessor provides a strong guarantee for real-time tasks' scheduling[1].

Distributed real-time scheduling:

Distributed real-time scheduling algorithms can be divided into two categories: I) the Generalized Rate Monotonic Scheduling (GRMS) 2) DSr based on pinwheel scheduling.

1) GRMS

GRMS is the use of RMS in distributed real-time system. It expands some basic concepts of RMS such as schedulability and preemption. At the same time, some new concepts such as system consistency are induced.[1]

2) DSr

In some real-time systems, tasks are executed on the way of distance constraint that is the intervals between same task's two executions in succession are less than or equal a certain time. Such real-time system is called Distance Constraint Tasks System (DCTS).[1]

The priority list is constructed recursively using the approach commonly referred to in the real time scheduling literature as the "Audsley approach"; it is also related to a technique introduced by Lawler. First determine the lowest priority job: Job Ji may be assigned the lowest priority if there is at least Pi (Xi) time between its release time and its deadline available when every other job Jj is executed before Ji for Pj

(Xi) time units (the WCET of job Jj according to the criticality level of job i). This can be determined by simulating the behavior of the schedule under the assumption that every job other than Ji has priority over Ji. [3].

CPU utilization is the total amount of work handle by the CPU. CPU utilization depends upon the task or processes. In Table 3 [4]processes P1, P3, P4, P5 and P6 contain the load < 10 or load ≥ 10 so the five processes start their execution because of their release time is earlier than other processes. Table 2 shows the execution of an Adaptive algorithm and the remaining five processes are executed when these process are release for execution. It depends upon their release time. [4]

Cache size MB	Algorithms % Hit Ratio Comparison						
	CAP	*RACE*	*ARC*	*LRU*	*LIRS*	*MQ*	*2Q*
8	44	54	43	43	44	43	42
16	45	54	44	43	44	43	46
32	48	54	45	44	46	44	44
64	50	54	46	45	45	45	45
128	59	97	46	46	77	54	75

TABLE III. HIT RATIO COMPARISON

VI. PROPOSED METHODOLOGY

The Hierarchical Scheduling Framework (HSF) has been introduced as a design-time framework to enable compositional schedulability analysis of embedded software systems with real-time properties. As real-time systems become complex and require high-performance, Hierarchical Real-Time Scheduling Framework has been investigated to schedule real-time tasks to meet their deadlines by sharing resources hierarchically under various scheduling algorithms. Scheduling algorithm dynamically adapts by making use of EDF and ACO algorithm. To manage under loaded and overloaded condition adaptive scheduling (AC) algorithm switched from EDF to ACO algorithm. The status of the process is changed from waiting to running depend upon the release time. Switching is performed between the processes it depends upon the load of each process. If the load, of the process change then the algorithm also changes.

Process is switch to ACO algorithm depending upon the following condition:

- Load of process is greater than the given CPU load.
- The process count.,

so a new methodology is proposed to use of adaptive scheduling algorithms in a Hierarchical Real-Time Scheduling Framework along with priority based algorithm in preemption mode. In proposed methodology scheduling initially starts with EDF scheduling algorithm but in case of tie occurs for providing the service to processes having earliest deadline, service provides to the process having highest priority. Also proposed methodology is implementing the pre-emptive mode in which current process is pre-empted to provide service to the process having earliest deadline so that better success rate in reduction of deadline miss can be achieved.

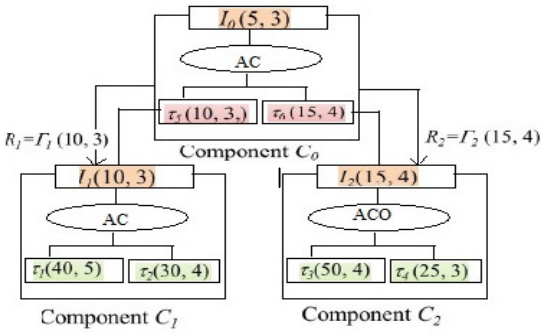

Fig 4. Proposed Scheduling Framework

OUTCOME POSSIBLE RESULT

An Adaptive algorithm removes the drawback of EDF and ACO algorithm and schedules more process than the EDF and ACO algorithm. Figure 5 [4]shows the Success Ratio and CPU utilization by using an Adaptive algorithm when system is overloaded.

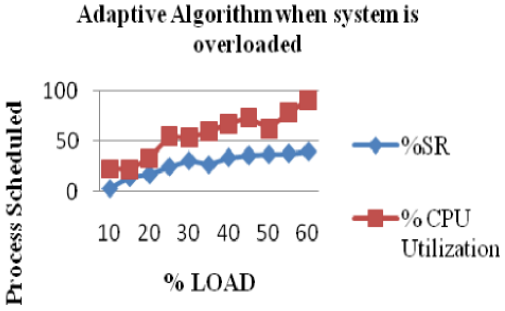

Fig. 5. Success Ratio and CPU Utilization of an Adaptive Scheduling Algorithm.

So by applying proposed framework the result will get improved at optimum level.

VII. CONCLUSION

This paper exhausted different Scheduling techniques used in the real time systems. It is very important and useful to analyse all the techniques for future purposes and searching of new techniques. The study on various methodologies applied for Scheduling of real time systems and various research issues in this field of study. There are different scheduling techniques studied in this paper, it concentrated on obtaining a better success ratio for process scheduling in real time systems. The propose method will provide better success ratio by the use of adaptive scheduling algorithm in hierarchical Real-Time Scheduling Framework.

VIII. FUTURE SCOPE

In the future, research efforts must be devoted to implementation of this framework for centralized multi-processor scheduling and distributed scheduling.

REFERENCES

[1] Li Jie, Guo Ruifeng, and Shao Zhixiang," The Research of Scheduling Algorithms in Real-time System", *IEEE 201O International Conference on Computer and Communication Technologies in Agriculture Engineering*, DOI: 978-1-4244-6947-5, PP. 333-336.

[2] Jongsoo Hyun,and Kyong Hoon Kim," Fault-Tolerant Scheduling in Hierarchical Real-Time Scheduling Framework" *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, DOI: 10.1109/RTCSA.2012.45,PP. 431-436.

[3] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Haohan Li,Alberto Marchetti-Spaccamela, Nicole Megow, and Leen Stougie, "Scheduling Real-Time Mixed-Criticality Jobs", ", *IEEE Transactions on Computers, VOL. 61, NO. 8,*, PP. 1140-1152, AUGUST 2012.

[4] Urmila Shrawankar, Ashwini Meshram, Reetu Gupta, and Jashweeni Nandanwar, "Pattern Based Real Time Disk Scheduling Algorithm for Virtualized Environment", *2013 IEEE Sixth International Conference on Emerging Trends in Engineering and Technology,* DOI: 10.1109/ICETET.2013.51, PP. 177-182