VFAST Transactions on Software Engineering
http://vfast.org/index.php/VTSE@ 2013 ISSN 2309-3978
Volume 1, Number 2, July-August 2013
pp. **07—11**

REVIEW OF DIFFERENT APPROACHES FOR OPTIMAL PERFORMANCE OF MULTI-PROCESSORS

UMM-I-AIMAN & SHER AFZAL KHAN
[1]Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan
aiman4822@yahoo.com, sher.afzal@awkum.edu.pk

Received June 2013

ABSTRACT. *We reviewed the literature used for optimal performance of multi-processor, we study different approaches in this paper. They include rate monotonic, deadline monotonic, and Earliest deadline first Algorithm. These approaches are basically used for real time scheduling systems .The problem of inconsistencies occurring in these algorithms such as those tasks whose task period is less but if not executed does not matter and whenever they are scheduled under rate monotonic scheduling algorithm the time consumed by CPU in scheduling the tasks is spent unnecessarily.*

Keywords*:* Real time tasks; Rate monotonic; Deadline monotonic; Priority, deadlines; Task periods

1. **Introduction.** A uniprocessors system is that system having only one central processing unit for the execution of tasks. In uniprocessing all the processing tasks are sharing the single central processing unit. In this the system executes one process at a time and takes the next job when the scheduled process is completed. Burah et.al., [1][2] deals with scheduling algorithms on uniprocessors for preemptive, non-preemptive and complex tasks. Alan Burns et.al., [6] describes problems on uniprocessors and they can be removed by dynamically assigning priority to the tasks. In Strosnider [10] it describes that periodic real time systems having regular arrival times and hard deadlines (a hard work, with the understanding that the deadline will be extended.) while the periodic have deadline, which is carved in stone, a soft deadline provides authors with a target date stop submitting their irregular interval and soft deadlines. This phenomena works fine when the tasks are less in number but as the number of tasks increases, it creates halting and slowing a system. To overcome such problems the concept of multiprocessing came.

The multiprocessor is a tightly coupled system having two or more CPU's, but they are all sharing the same memory and peripheral's in order to execute the tasks one after another . The term Multiprocessor refers to the hardware architecture that allows *multiprocessing*. It plays a very important role in multitasking; however, it has been associated with problems, like, load balancing, collision of tasks and heating up of the CPU. There exists some documented work which shows solutions towards the problems. For example Bier, George[20]. study the contention effect on a single semaphore (shared) for the protection of critical sections of Multiprocessors. Cybenko[21] describes diffusion schemes for dynamic load balancing on message passing on multi-processor environment. Jensen, E.D [16] described the following four paradigms: static table-driven scheduling, static priority preemptive scheduling, dynamic planning-based scheduling, and dynamic best effort scheduling. Static table-driven scheduling performs static analysis and the resulting scheduling used for the purpose to show which task would start its execution but this decision is taken at runtime. The static priority preemptive scheduling performs static scheduling but unlike static driven there is no need of construction of scheduling first but at runtime tasks of higher priority are executed. The dynamic planning-based scheduling in this scheduling main concept is of feasibility and also those tasks will be

executed which are feasible and this all is done at runtime . In the dynamic best effort scheduling there is no checking of feasibility but the system is in the struggle to meet a deadline, also there is no guarantee and tasks may be terminated at runtime as well Bini, Enrico et.al., [5] describes a variety of scheduling policies for embedded real time systems. Rate monotonic scheduling theory fully described that how rate monotonic theory can be applied to practical systems and there is also a great knowledge of reusing hardware and software which are existing in the development system. They review the theory and its implications for Ada (The ADA tasking model is all about of priority driven scheduling system where there is a great control of concurrency occurring in the tasks Liu, C. L., & Layland Burns, A., & Wellings [16][12] In Bini, E., & Buttazzo [18] it is describe real time systems with digital processing. C. U LIU describes that a process can be fully utilized where there is an assignment of priority but that assignment will be dynamic and on the basis of their current deadlines. Willebeek [21] describes that how load can be balanced on parallel computers. Also those strategies are discussed which a parallel system has to face such as when there is dynamic load of the task is occurring on a single processor in a multicore environment.

2. Algorithms for Optimal scheduling.

In this section we review algorithms used for optimal scheduling Rate montonic, Deadline monotonic and Earliest deadline first**.**

2.1. **Rate Monotonic.** Rate-monotonic scheduling is used for real time system for static class scheduling i.e., static priorities are assigned on the basis of cycle duration so shorter the job cycle higher will be priority vice versa Buttazzo, Bini, Enrico [4] [5]. In rate monotonic each periodic process completes within its slot, there is no interposes dependencies each process need the same amount of CPU time, non–periodic process have no deadlines, preemption happens instantly with no overhead. This algorithm generally describes the concept of rate of occurrence, lower the priority higher is the rate of occurrence, and on the bases of this rate priorities are assigned to the tasks. Buttazzo [5]. Rate-monotonic is proved to be the optimal static priority for real-time task scheduling. Amongst the class of static priority scheduling schemes, the priority assignment of rate monotonic is the best one Selic[7],.

2.2. Deadline-monotonic algorithm: It is that approach used for fixed pre-emptive scheduling environment. The deadline monotonic priority task CPC algorithm according relative to deadlines (an *absolute deadline* is the moment in time at which the response must be completed.) by Andes lay [19]. So shorter the response completion time higher will be priority. But the rate monotonic and deadline monotonic give the same results when the relative deadline of every task is being matched with its period by Andes lay [19]. But when the relative deadline is arbitrary then it produces feasible scheduling. Also there is concept of ordering and this ordering is assigned to the process are inversely proportional to the length of the deadline so shorter the length higher will be the priority of the task [3]. This defaults to Priority Ordering a rate monotonic period=When Ordering deadlines. Deadline monotonic priority assignment is an optimal scheme (static) for that 's processes hare critical instant "The inverse-deadline priority assignment is an optimal priority assignment for one processor [8].

2.3. Earliest deadline first:

EDF is a dynamic scheduling algorithm used in real-time operating systems to place processes in a priority queue. In EDF, there is occurring of some scheduling events like (task finishes newly task arrived) then that task will be in the queue having closest deadline. EDF is an *optimal* scheduling algorithm on preemptive uniprocessor, it treats collection of all tasks independently, each and every task is summarized on the bases of their arrival time. Further, EDF schedules these collection of tasks to complete their deadline [25]. These scheduling periodic processes have deadlines equal to their periods; **EDF** has a utilization bound of 100%. Thus, the schedulability test for **EDF** by Han, C. C et.al., [26]

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1,$$

Where the $\{C_i\}$ are the worst-case computation-times of the n processes and the $\{T_i\}$are their respective inter-arrival periods (assumed to be equal to the relative deadlines). In EDF some undesirable deadline interchanges may occur with EDF scheduling. Let suppose when a shared resource is accessed by processes using critical sections within a process (to prevent it from being pre-empted by another process with an earlier deadline waiting for access to the same shared resource) then it is responsibility of scheduler to assign EDF

(temporarily) to the tasks waiting for the resource in the critical section.

Now consider 3 periodic processes scheduled using EDF, the following acceptance test shows that all deadlines will be met. Where, C is Process Execution time of Periods T. In table 1 we take some values execution time & time period and derive its result.

Table #1. Example of EDF: 3 processes are taken w.r.t there execution time and time period

processes	Execution time=c	Period=t
P1	2	6
P2	3	7
P3	4	8

By EDF, utilization is: 1/8+2/5+4/10=0.925=92.5%. As it is dynamic priority assignment so here any number of task can be accommodated and scheduled. As system is scheduled fully but there is occurrence of some unoccasional deadlines then at that time scheduler is responsible to assigns EDF to the tasks in critical systems. And when this critical time for the resource is too long then that process must be exit from that critical section Devi, U. C.[27].

In the method, processes are sorted in order according to their deadlines if a new process is arrived then it could be inserted first in the list as compared to that process having later deadlines with less time search[28].

Comparison of approaches: In the following table there is a summarized description of the approaches we used in our paper .These approaches are compared as well according to there task scheduling time constraint deadline and so on are in this table :

Table 2. Comparison of approaches:

Criteria	**Rate monotonic**	**Deadline monotonic**	**Earliest deadline First**
Task importance	Equal importance to all Tasks [24]	Equal importance to all tasks [24]	Task with earlier deadline is assigned high priority.by [3]
Scheduling criteria	Task period [24]	Task deadline [24]	Earlier deadline [3]
Task scheduling	Tasks are arranged in ascending order based on task period [24]	Tasks are arranged in ascending order based on task deadline [24]	No arrangement in any order (SJF)
Time difference	Task period does not vary with time [24]	Task deadline does not vary with time [24]	Deadline vary with time [9]
Type of scheduling	Static scheduling by [8]	Static scheduling by [8]	dynamic scheduling by [8][14]

Conclusion : The approaches we studied in this paper used for both single as well as for multiprocessors. The paper also reviewed some existing problems in the single processor as (task collision, delay in processing and problems of inconsistencies etc.). Further, we reviewed different algorithums used for multiprocessor which has concurrent approach shows more efficient processing than that of uniprocessor. The multiprocessor works efficiently for limited tasks as well as when the tasks are are increases which can cause inefficiency for single processor. Here in this case speed of processing is enhanced as compared to uniprocessors. We analysed that rate monotonic and deadline monotonic scheduling are static while earliest deadline first scheduling algorithm is dynamic.

REFERENCES

[1]. Baruah, S. K., Mok, A. K., & Rosier, L. E. (1990). Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Real-Time Systems Symposium, 1990. Proceedings, 11th* (pp. 182-190). IEEE.

[2]. Baruah, S. K., Rosier, L. E., & Howell, R. R. (1990). Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, *2*(4), 301-324.

[3]. Okuyan, E., & Kayayurt, B. (2012). Earliest deadline first scheduling algorithm and its use in ANKA UAV. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st* (pp. 8B1-1). IEEE.

[4]. Buttazzo, G. C. (2005). Rate monotonic vs. EDF: judgment day. *Real-Time Systems*, *29*(1), 5-26.Jensen, E. D., Locke, C. D., & Tokuda, H. (1985, December). A Time-Driven Scheduling Model for Real-Time Operating Systems. In *RTSS* (Vol. 85, pp. 112-122).

[5]. Bini, Enrico, and Giorgio C. Buttazzo. "Schedulability analysis of periodic fixed priority systems." *Computers, IEEE Transactions on* 53, no. 11 (2004): 1462-1473.

[6]. Burns, A., & Wellings, A. J. (2001). *Real Time Systems and Their Programming Languages: Ada 95, Real-time Java and Real-time POSIX*. Pearson Education.

[7]. Selic, B. (1996). Tutorial: real-time object-oriented modeling (ROOM). In *Real-Time Technology and Applications Symposium, 1996. Proceedings. 1996 IEEE* (pp. 214-217). IEEE

[8]. Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, *20*(1), 46-61.

[9]. Sprunt, B., Sha, L., & Lehoczky, J. (1989). Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, *1*(1), 27-60.

[10]. Strosnider, J. K., Lehoczky, J. P., & Sha, L. (1995). The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *Computers, IEEE Transactions on*, *44*(1), 73-91.

[11]. Tindell, K. W., Burns, A., &Welling's, A. J. (1992). Allocating hard real-time tasks: an NP-hard problem made easy. *Real-Time Systems*, *4*(2), 145-165.

[12]. Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, *20*(1), 46-61.

[13]. Lea, D. (2000). *Concurrent programming in Java: design principles and patterns*. Addison-Wesley Professional.

[14]. Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, *20*(1), 46-61.

[15]. Kuo, S. M., Lee, B. H., & Tian, W. (2006). *Real-time digital signal processing: implementations and applications*. Wiley. Com.

[16]. Jensen, E. D., Locke, C. D., & Tokuda, H. (1985). A Time-Driven Scheduling Model for Real-Time Operating Systems. In *RTSS* (Vol. 85, pp. 112-122).

[17]. Ramamrithm, K., & Stankovic, J. A. (1994). Scheduling algorithms and operating systems support for real-time systems. *Proceedings of the IEEE*, *82*(1), 55-67.

[18]. Bini, E., & Buttazzo, G. C. (2004). Schedulability analysis of periodic fixed priority systems. *Computers, IEEE Transactions on*, *53*(11), 1462.

[19]. Audsley, N. C., Burns, A., Richardson, M. F., & Wellings, A. J. (1991). Real-Time scheduling: the deadline-monotonic approach. In *in Proc. IEEE Workshop on Real-Time Operating Systems and Software.*

[20]. Bier, George E., and Mary K. Vernon. "Measurements and prediction of contention in multiprocessor operating systems with scientific application workloads." In *Proceedings of the 2nd international conference on Supercomputing*, pp. 9-15. ACM, 1988.

[21]. Cybenko, G. (1989). Dynamic load balancing for distributed memory multiprocessors. *Journal of parallel and distributed computing*, *7*(2), 279-301.

[22]. Willebeek-LeMair, M. H., & Reeves, A. P. (1993). Strategies for dynamic load balancing on highly parallel computers. *Parallel and Distributed Systems, IEEE Transactions on*, *4*(9), 979-993.
[23]. George, L., Rivierre, N., & Spuri, M. (1996). Preemptive and non-preemptive real-time uniprocessor scheduling.
[24]. C. Y and Ramesh, R. (2010) A new scheduling algorithm for real time system, (international journal of computer and electrical engineering, vol.2, no.6, 1793-8163).
[25]. Okuyan, E., & Kayayurt, B. (2012). Earliest deadline first scheduling algorithm and its use in ANKA UAV. In Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st (pp. 8B1-1). IEEE.
[26]. Han, C. C., & Tyan, H. Y. (1997). A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE (pp. 36-45). IEEE.
[27]. Devi, U. C. (2003). An improved schedulability test for uniprocessors periodic task systems. In Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on (pp. 23-30). IEEE.
[28]. P alencia Gutiérrez, J. C., Gutiérrez García, J. J., & González Harbour, M. (1998). Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems. In Real-Time Systems, 1998. Proceedings. 10th Euromicro Workshop on (pp. 35-44). IEEE.