International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)

Impact Factor: 5.22 (SJIF-2017), e-ISSN: 2455-2585

Volume 4, Issue 10, October-2018

The Comparative Analysis of Scheduling Approaches in Real-Time Systems

Jayna Donga [1], Dr. M.S.Holia [2]

[1]Ph.D. Scholar, Gujarat Technological University, Ahmedabad, Gujarat

[1]jdonga@mbict.ac.in

[2]msholia@bvmengineering.ac.in

Abstract. The Real Time Operating System (RTOS) supports applications that meet deadlines in addition to providing logically correct results. In multiprocessing operating system for the applications need to meeting of time deadlines and functioning in real time constraints. The quality of real-time scheduling algorithm has a direct impact on real-time system's working.The scheduling algorithms mainly studied in this paper are Earliest Deadline First, Rate Monotonic, Deadline Monotonic, Least laxity First, Group Earliest Deadline First for periodic task.We observed that the choice of a scheduling algorithm is important in designing a real-time system.

Keywords: Deadline,Scheduler, Slack-Time,Real-Time Systems, Period, EDF,RM, DM, GPEDF, GEDF, LLF.

1 INTRODUCTION

Real time system must respond to externally generated inputs within a specified period to avoid failure. The deadline of a task is the point in time before which the task must complete its execution [1]. Real-Time system can be divided into three categories; 1) Soft Real-Time System:In this type of deadline, task could miss some deadline and the system could still work correctly. Reservation systems is one of the example of soft deadline.Its result is having less useful,2) Firm Real-Time System:This deadline is one in which the results come after the deadline is missed is of no usefulness. Infrequent deadline misses are tolerable.Its result is having zero percent utilization if created after deadline,3) Hard real-Time System:If task miss some deadline, then catastrophe results will occur, such type of deadline is known as hard deadline. The system which are performing critical applications like air traffic control go under this category.

The application of real time systems can be found in Robotics, Pacemakers, Chemical Plants, Antimissile Systems, and Embedded Systems etc. to name a few [2]. There are three kinds of real-time tasks, depending on their arrival pattern: periodic task: Periodic tasks will be executed at every fixed time period. Normally, Aperiodic task: aperiodic tasks will be executed at any random time constraints and would not have pre-defined time duration and Sporadic task: Sporadic tasks are combination of both periodic and Aperiodic, where in, the executing time is aperiodic but the executing rate is periodic in nature. The time constraints are usually a deadline. Scheduling mechanism is the important concept of a computer system, it is the strategy by which computer system will decide which task should be executed in which order. Scheduling algorithm for uniprocessor systems must guarantee to apportion the enough time to all the system task at specific purposes of time that they can meet their deadline as far as possible.

The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible when we consider soft real-time system [3]. To achieve this goal, vast researches on real-time task scheduling have been conducted. Real-time scheduling can be divided into two categories: Static and Dynamic. In static algorithm al priorities are assigned at design time and those priorities is remains constant for the life time of a task. Dynamic algorithms assign priorities at runtime, based on execution parameters of tasks. Dynamic

scheduling can be either with static priority or dynamic priority. Rate Monotonic [4] and Deadline Monotonic [5] are examples of dynamic scheduling with static priority. EDF [4] (Earliest Deadline First) and LLF [6] (Least Laxity First) are examples of dynamic scheduling with dynamic priority. EDF and LLF algorithms are optimal under the condition that the jobs are pre-emptive, there is only one processor and the processor is not overloaded [7]. But the limitation of these algorithms is, their execution diminishes exponentially if the system becomes overloaded. This paper presents comparative analysis of various well-known task scheduling approaches.

2 LITERATURE SURVEY

The next section is dedicated to the research and researchers who have played an important role for the domain of real-time operating systems and especially in the area of real-time scheduling.

Habibah Ismail et.al.[1] has presented a hybrid scheduling algorithm for weakly hard real-time tasks that can be used in a multiprocessor environment in 2017. For scheduling the real-time jobs in a multiprocessor environment, there are two main approaches; 1) Partitioning scheduling 2) Global scheduling. Though partitioned method having tolerable overhead, it doesn't provide any kind of guarantee of being optimal. While in the case of the Global method, such kind of guarantee can be provided, but it has substantial overhead. In this paper, the researcher had proposed an unconventional Scheduling tactic to take advantage of both the partitioning scheduling strategy and the global scheduling strategy. Hybrid algorithm having less overhead as it limits the number of context switches of a task and to increase the schedulability by defining an efficient allocation of jobs to the CPU. They observed that proposed algorithm gives optimal schedulability with minimum overhead.

Hoon Sung Chwa et.al.[6] has presented an Optimal Real-Time Scheduling on Two-Type Heterogeneous Multicore Platforms in 2015. This paper uses the global (fully-migrative) approach to two type heterogeneous multicore scheduling like ARM's big. LITTLE.In this paper they extended one of the simplest optimal scheduling algorithms for identical multicore platforms toward two type heterogeneous scheduling.First optimal two type heterogeneous multicore scheduling algorithm called hetro-wrap that has the same complexity o(n) as in identical(homogeneous) multicore case.Recently new optimal algorithms such as B-fair,Run,U-EDF and QPS were proposed for identical multicore platforms with aim of reducing number of preemptions and migrations.**1)**As a future work we can extend those advanced scheduling techniques to two-type heterogeneous scheduling with the additional consideration on two types of migration showing different costs ,aiming at reducing the overall preemption and migration.

Nasro Min-Allah et. Al.[13] has presented a comparative study of rate monotonic schedulability tests in 2011.With the increased penetration of real-time systems into our surroundings,the selection of an efficient schedulability test under fixed priority system from existing results, has become a matter of primary interest to real-time system designers. The need for a faster schedulability tests becomes more prominent when it applies to online systems, where processor time is a sacred resource and it is of central importance to assign processor to execute tasks instead of determining system schedulability. Under fixed priority nonpreemptive real-time systems, current schedulability tests (in exact form) can be divided into: response time based tests, and scheduling points tests. The aim of this paper is to assist the system designers in the process of selecting a suitable technique from the existing literature after knowing the advantages and disadvantages associated with these tests. They have shown the mechanism behind the feasibility tests, theoretically and experimentally.This paper proves by experimental results that response time based tests are faster than scheduling points tests,which make the response time based tests an excellent choice for online systems.

Sangchul Han et. Al.[11] has presented the Predictability of Least Laxity First Scheduling Algorithm on Multiprocessor Real-Time Systems in 2006. A priority-driven scheduling algorithm is said to be *start time (finish time) predictable* if the start time (finish time) of jobs in the schedule where each job executes for its actual execution time is bounded by the start times (finish times) of jobs in the schedules where each job executes for its maximum/minimum execution time. In this paper, they studied the predictability of a job-level dynamic priority algorithm, LLF (Least Laxity First), on multiprocessor real-time systems. They have shown a necessary and sufficient condition for a priority-driven algorithm to be start time (finish time) predictable. Then, in LLF scheduling, they shown that both the start time and the finish time are predictable if the actual execution times

cannot be known. However, solely the finish time is predictable if the actual execution times can be known.

Li, Q. & Ba, W et. al.[8] has presented the scheduling algorithm, which considers group priority earliest deadline first in 2012. In most of the priority scheduling algorithms, the assumption is taken that priority levels are unrestricted, but in some of the cases in which task set wants priority levels more than the system's capacity, so the same priority level will be assigned to the several jobs. To address this issue, the new scheduling algorithm GP-EDF has been designed in that group-wise priority is assigned, and the group which is having the earliest deadline will contain the highest priority and executed first. The proposed approach provides Schedulability test to form a task group and the jobs within one group may change their execution order arbitrarily without dropping the Schedulability. In this paper, new proposed algorithm GPEDF was compared with the traditional algorithm EDF and the gEDF. The results show that the proposed algorithm is having less switching overhead, the shortest average response time, and it requires very few priority levels.

3 SCHEDULING ALGORITHMS

Nowadays, we are using computers are multi-tasking and multi-processing systems. We mean by multi-tasking is the system can run multiple tasks simultaneously. A Computer system is full of resources, which may be either software or hardware. These resources will be shared among all the tasks into the systems. To take a decision about which resource will be given to which task at which instance of time is an important task in any computer system. The module of OS called scheduler is accountable for taking judgment about resource allocation and De-allocation during the task's execution.so to design an efficient scheduling algorithm is an important operating system design issue. The Goals of any scheduling algorithm is to distribute equal load among all the processors, maximize resource utilization, minimize response time, and maximize throughput [9]. As especially whenever we are talking about the real-time operating system, then important scheduling criteria is meeting the deadlines for all the tasks into the system.

3.1 Fixed Priority Scheduling Algorithms- RM and DM

1) Rate Monotonic (RM) [5, 7, 8, 9, 10,11]: The Rate Monotonic algorithm is an optimal static priority algorithm with preemption. It schedules periodic tasks with deadline equal to period. Each task is assigned a fixed priority inversely
proportional to period. This implies that a task with a shorter period has a higher priority. A set of n tasks will always meet their deadlines when scheduled according to the rate monotonic approach if,

$$\text{Utilization } (U) = \sum_{i=1}^{n} Ci/Pi \leq n(2^{1/n} - 1)$$ where Ci = worst-case computation time, Pi = period

2) Deadline Monotonic (DM) [12,13]: The Deadline Monotonic algorithm is an optimal static priority algorithm with
preemption. It schedules periodic tasks with deadline less than or equal to period. Each task is assigned a fixed priority inversely proportional to relative deadline. This implies that a task with a shorter relative deadline has a higher priority. The rate monotonic algorithm is a special case of the deadline monotonic approach when deadline is equal to the period.

3.2. Dynamic Priority Scheduling Algorithm- EDF

1) Earliest Deadline First (EDF) [5,8]: The Earliest Deadline First algorithm is an optimal dynamic priority algorithm
with preemption. It schedules periodic tasks with deadline equal to period. Each task is assigned a fixed priority inversely proportional to absolute deadline. This implies that a task with a shorter absolute deadline has a higher priority. A set of n tasks will always meet their deadlines when scheduled according to the earliest deadline first approach if,

$$\text{Utilization } (U) = \sum_{i=1}^{n} Ci / Pi \leq 1$$ where Ci = worst-case computation time, Pi = period

2) Least Laxity First scheduling Algorithm (LLF)[9,19,20]:

LLF is another optimal dynamic-priority scheduling algorithm. The laxity of a process is defined as the deadline minus remaining computation time. The laxity of a job is the maximal amount of time that the job can wait and still meet its deadline. The algorithm gives the highest priority to the dynamic job with the littlest laxity. Then the job with the highest priority is executed. While a process is executing, it
can be preempted by another whose laxity has less than that of running process. A problem arises with this scheme when two processes have similar laxities. One process will run for a short period while and then get preempted by the other and
vice versa. Hence, numerous context switches happen in the lifetime of the processes. The least laxity first algorithm is an optimal scheduling algorithm for systems with periodic realtime tasks [10].

3) Group Earliest Deadline First (G-EDF)[6,13,17]:

gEDF was developed for improving the success ratio of EDF during overload condition of soft real time multi-media application. The initiator pioneered the idea of group scheduling, where jobs with near deadlines were group together using an algorithm. After grouping jobs within a group are schedule using shortest job first sched-uling [9, 11].Group range parameter (Gr) determines which job gets into which group. It is simply a percentage value of the job at the head of a queue's absolute deadline. Mathematically it is defined as

gEDF Group = {t k | t k ∈ QgEDF , d k - d1 = d1 Gr ,1< = k, m< = | QgEDF |} (1)

in which:

- d1 is the dynamic deadline of the first job in the group
- QgEDF is a queue for gEDF and
- | QgEDF | represents the length of queue
- m is the number of all ready jobs in a system [9]

4 CONCLUSION

The survey about the various Real-Time scheduling approaches has been covered in this paper. The classification of scheduling algorithms is done based on various parameters. The paper mainly categories real-time scheduling algorithms into two categories, like dynamic real-time scheduling algorithms having static priority and dynamic real-time scheduling algorithms having dynamic priority. It includes the study of very renowned real-time schedulers like RM, DM, EDF, LLF,G-EDF and the comparison between all these algorithms.we have analyzed the advantages and disadvantages of each based on various performance measure parameters. Basically the existing algorithms give better performance for the uniprocessor system, but nowadays, we are working on a multiprocessor environment so one can focus on multiprocessor Real-Time Systems and can design an efficient multiprocessor scheduling algorithm.Table-1 shows the comparitive study of various real-time scheduling algorithms.

Table-1. Comparative Study of different Scheduling Approaches for Real-Time System

Tech-niques / Performance Matric	**RM**	**EDF**	**DM**	**LLF**	**GEDF**
Implementation	Easiest	Complex	Easy	Complex	Complex
Priority Type	Static	Dynamic	Static	Dynamic	Dynamic
CPU Utilization	Low	Full Utili-zation	High as compared to RM	Full Utilization	Full Utilization
Scheduling Parameters	Period of task	Deadline	Relative Deadline	Laxity (slack time)	Group prior-ity and SJF within Group

Effectiveness	Good transient overload handling capacity only for high priority tasks	Efficient in under loaded condition	Good transient overload handling capacity only for high priority tasks	Efficient	Efficient in Non-Preemptive environment
Limitations	a. Only supports periodic tasks b. Not optimal when task periods and deadlines are different	Not efficient under overloaded system	optimal algorithm for uniprocessor only with static priority	Can't handle transient overload condition optimally	Good for Non-preemptive environment only
Pre-emptive/Non-Pre-emptive	Non-preemptive	Preemptive	Preemptive	Preemptive	Non-Preemptive

REFERENCES

[1] R. L. Panigrahi and M .K. Senapaty, "Real Time System for Software Engineering: An Overview", Global Journal for Research Analysis, Vol. 3, Issue 1, pp. 25-27, January 2014.

[2] Jane W.S. Liu, Real-Time Systems , Pearson Education, India,pp. 121 & 26, 2001.

[3] Mehrin Rouhifar and Reza Ravanmehr, "A Survey on Scheduling Approaches for Hard Real-Time Systems", International Journal of Computer Applications (0975 – 8887) Volume 131 -No.17, December 2015.

[4] C. Liu and James Leyland, January 1973, "Scheduling algorithm for multiprogramming in a hard real-time environment", Journal of the Association for Computing Machinery, 20(1): 46-61.

[5] J.Leung and J. Whitehead, "On the complexity of fixed- priority schedulings of periodic, real-time tasks", Performance Evaluation 2(4):237-250 December 1982.

[6] Hoon Sung Chwa, Jaebaek Seo, Jinkyu Lee, Insik Shin, "Optimal Real-Time Scheduling on Two-Type Heterogeneous Multicore Platforms", Published in 2015 IEEE Real-Time Systems Symposium.

[7] A. Wiese, V. Bonifaci and S. Baruah, "Partitioned EDF scheduling on a few types of unrelated multiprocessors", Springer, Real-Time Syst, vol. 49, pp:219–238, 2013.

[8] Li, Q. & Ba, W, "A group priority earliest deadline first scheduling algorithm", Frontiers of Computer Science October 2012, Volume 6, Issue 5, pp 560–567.

[9] J. Rosen, P. Eles, A. Andrei, Z. Peng, "Bus access optimization for predictable implementation of realtime applications on multiprocessor systems-on-chip", In: Proceedings of the 28th IEEE real-time system symposium, 2007.

[10] German standard DIN 66243-2, "Programmiersparche PEARL90", Beuth, 1998.

[11] Sangchul Han, Minkyu Park, "Predictability of Least Laxity First Scheduling Algorithm on Multiprocessor Real-Time Systems", in International Conference on Embedded and Ubiquitous Computing EUC 2006: Emerging Directions in Embedded and Ubiquitous Computing pp 755-764 As a Part of

the Lecture Notes in Computer Science book series (LNCS, volume 4097)

[12] J. Lehoczky, L. Sha and Yv Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior", IEEE International symposium on real time system, pp. 166-171, 1989.

[13] Nasro Min-Allah,Samee Ullah Khan,Nasir Ghani,Juan Li · Lizhe Wang,Pascal Bouvry, "A comparative study of rate monotonic schedulability tests" , in Springer,02 February,2011.

[14] Jane W.S. Liu, Real-Time Systems , Pearson Education, India,pp. 121 & 26, 2001.

[15] J. Liu, Real-Time Systems, Pearson Education, 2000.

[16] K. Kotecha and A. Shah,"ACO based dynamic scheduling algorithm for real-time operating system", Sent to AIPR-08, Florida, 2008.

[17] G.Saini, "Application of fuzzy logic to real-time scheduling", Real TimeConference, 14th IEEE- NPSS, 2005.

[18] Arezou Mohammadi and Selim G. Akl," Scheduling Algorithms for Real-Time Systems", Technical Report No. 2005-499, July 15, 2005.

[19] Zahereel Ishwar Abdul Khalib , Badlishah R. Ahmad and Ong Bi Lynn Ong, "High deadline meeting rate of non- preemptive dynamic soft real time scheduling algorithm",296301,DOI:10.1109/ICCSCE.2012.648715,
2012 IEEE.

[20] G. Yao, R. Pellizzoni, S. Bak, E. Betti and M. Caccamo, "Memory-centric scheduling for multicore hard real-time systems", Springer, Real-Time Syst, vol. 48, pp:681–
715, 2012.

[21] W. Li, K. Kavi, and R. Akl, "A non-preemptive scheduling algorithm for soft real-time systems", Computers and Electrical Engineering, vol.33, no. 1, pp.
12–29, 2007.

[22] A.Mok and M.Dertouzos, 1978, "Multiprocessor scheduling in a hard real-time environment",7th Texas Conference on Computing Systems.