JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Study and Performance analysis of different multiprocessor real time schedulers

[1]Ramappa Hiremani, [2]Siddesha K, [3]Kavitha Narayana B M
[1, 2] Department of Electronics and Communication Engineering
[3] Department of Electronics and Telecommunication Engineering
[1]Dr. Ambedkar Institute of Technology, Bengaluru, India

Abstract : A scheduling system for real time scenario consists of clock, processing hard-ware elements and clock. In real time systems, a task/process exhibits schedulability in which real time system accepts the tasks and does task completion based on the task deadline defined in the scheduling algorithm. To perform the real time scheduling, different scheduling algorithms were developed and are able to man-age the multiprocessor architectures. However, these algorithms are leading to new challenges in scheduling due to more complexity of the multiprocessors. Al-so, the scheduling algorithms cannot be evaluated without a real as well as complex implementation. This manuscript focused on the study and performance analysis of different multiprocessor real time schedulers. The performance analysis is carried out by using SimSo scheduling tool that extracts the scheduler class from python. In this, the scheduling algorithms like Deadline Partitioning (DP)-WRAP, EKG, Earliest deadline first (EDF) are studied and analyzed for their performance. From the analysis is observed that Load on EDF is much different than EKG and DP-WRAP as it meets deadlines. Most of the time, the schedulers like DP-WRAP and EDF are considered as partitioned scheduler most of the time and also DP-WRAP scheduler generates lot of migrations. Hence, EDF scheduler is considered as more efficient than DP-WRAP and EKG schedulers based on preemptions and migrations. However, some of the jobs are aborted due to dead-line misses**.**

IndexTerms **- Real time scheduling, Multiprocessor, Scheduling algorithms, Preemptions and Migrations.**

I. INTRODUCTION

The real time systems (RLS) are capable of carrying the real time process/tasks and are need to be executed immediately by considering the event as inevitable on the basis of priority. Generally, the tasks are meant to control particular events and reacts those events. The real time tasks are categorized as hard and soft RLS [1, 2]. A hard RLS (HRLS) is not allowed to skip any deadline and which can causes complete system failure. A HRLS can be safety critical as it can harm environment or persons upon missed deadline and system failure. Also, the HRLS should meet deadlines in time without any delay and not before defined time. However, the soft RLS (SRLS) can be allowed to skip deadline and hence system failure cannot take place but the performance can be reduced.

The short term schedulers in RLS to schedule the tasks is considered as more significant component as it minimizes the response time of each task than deadline handling. In case preemptive scheduler (PS) is used, the tasks in RLS need to wait till the respective task time slot ends. However in non-preemptive schedulers (NPS), the low priority task should wait till high priority tasks are completed that leads to longer wait time. Hence, to have better scheduling performance PS and NPS can be used together. The integration of PS and NPS be done by providing time based interrupts over the already running process. Also, the ready queue will be executed based on its higher priority to per preemption of ongoing process [3, 4]. This paper describes the study and performance analysis of different real-time scheduling policies of multi-processor. The paper is categorized into subsections like section-II (review of literature), section-III (Problem Statement), section-IV (Research Methodology), section-V (Results and discussion) and section-VI (Conclusion)

II. REVIEW OF LITERATURE

The existing scheduling strategies of multiprocessor in real-time are designed without the consideration of caches and their effect over the system behavior. A particular work of (Cheramy et al., 2018) introduced a flexible real time scheduling that offers controlling on job computation time. The author has considered different algorithms for comparative analysis and found that G-EDF is more promising in handling cache related delays [8] [9]. Similarly, if multiple processors share a cache then task execution affects already existing task running on different processor. However, the schedulers may generate many of the migrations, preemptions and may give rise to more rescheduling points to have better utilization of the processor (Devi and Anderson, 2005) [10]. Liu et al. [11] has presented a hybrid concept of scheduling algorithm (SA) to schedule the tasks and message. This scheduling algorithm sets the deadline for subtasks and a particle swarm optimization (PSO) algorithm is subjected to the subtasks so that optimization of control tasks can be achieved. The simulation outcomes suggest that scheduling algorithm and the PSO algorithm are fount more effective. The scheduling of real time tasks in the heterogeneous environments is more difficult task. Such problem is addressed in Li et al., [12] by using deadline constraint SA (DCSA). The analysis of the DCSA shows that it avoids the thrashing of scheduling system and

yields successful scheduling task ratio. The data dependent task re-scheduling mechanism is introduced in Xiaoqing et al., [13] where tasks are set based on the priority. The mechanism is tested in both homogenous and heterogeneous resources where it is observed improved scheduling length and least power consumption. A load balancing based real time scheduling is presented in Zhang et al. [14] where periodic tasks are considered in multiprocessor environment. The study yields stable and balanced load performance. Another work of Zhang et al., [15] have presented a heuristic algorithm for task scheduling to achieve high performance through task and list duplication concept.

A work towards resource attribute selection based task scheduling is found in Zhao et al. [16]. From obtained results it is found that execution throughput is improved and efficient resource utilization. The preemptive scheduling of the real time tasks are addressed in Wang et al. [17] by using the discrete-event system based control model. The outcomes suggest that it control model able to offer safe real time task scheduling. In Rincon et al. [18], have addressed real time scheduling with information principle to minimize the job migrations while Dong et al. [19] have mentioned general framework for soft computation of real time task that gives improvement in worst case task execution. Works of Sun et al., [20] have focused on minimization of energy consumption in real-time task scheduling with flow network based approach. In [21] authors have proposed a hybrid energy aware task scheduler for embedded computing systems. Also, some of the existing works like Martini et al., [22] have described peak load optimization, Huang et al., [23] given a fault tolerant scheduling mechanism, Doan et al., [24] have expressed task assignment mechanism and Lee et al., [25] priority based task assignment for Multi-Processor Real-Time Scheduling. The next section-III gives the findings of existing works and problem statement.

III. Problem Statement

Different scheduling algorithms were developed to perform the real time scheduling, and are able to manage the multiprocessor architectures. However, these algorithms are leading to new challenges in scheduling due to more complexity of the multiprocessors. Also, the scheduling algorithms cannot be evaluated without a real as well as complex implementation. From, the above literature it is observed that various real time scheduling of multiprocessor architectures are presented in recent past by focusing on new form of issues in handling multiprocessor architectures in terms of complexity. However, the existing techniques are lacking with the difficulty in real time implementation as well as complexity concerns leading to the degradation of scheduling efficiency and system failure. Thus, problem statement is that "there is a need of study and analysis of various schedulers in real time scenarios".

IV. System Design

The study and analysis of the various schedulers is performed by using SimSo which is an open access tool that inherits the scheduler class from python. Following Fig.1, indicates the mutual interaction among the main classes of SimSo. The SimSo design is inspired with the real time systems having processors, timers, jobs etc. Every object in this system simulates respective behavior on the system i.e., task→releases→jobs, jobs→follows→task code execution, timer → to→launch→ method over processor etc. The instances of the processor perform the simulation of both the OS and processors (which makes processor as central unit of simulation).

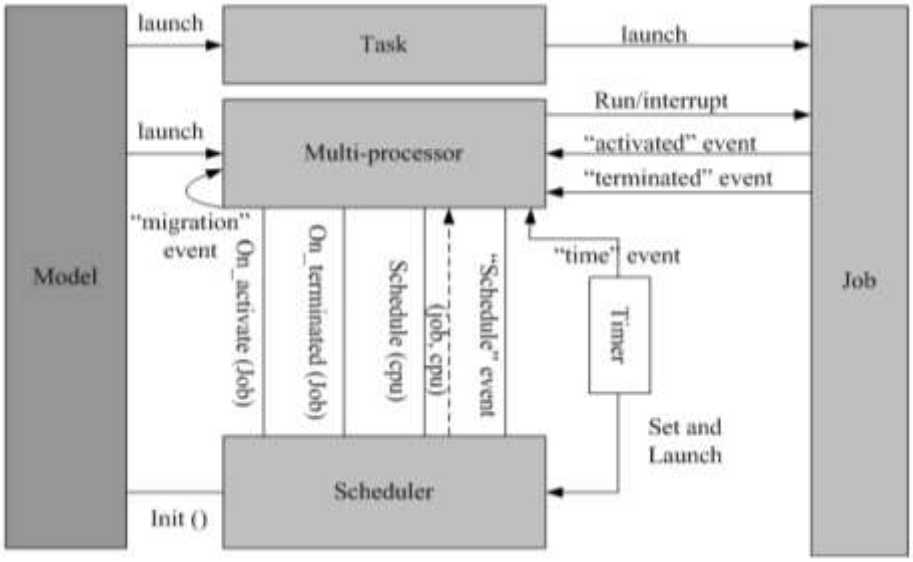

Figure.1. Mutual interactions among main class instances

Every processor of the system does job execution/interruption to execute scheduler method. The scheduler is not considered as active process hence it is a part of OS while its methods are called. The model object does the simulation that considers configuration object. If the model method is called then the described objects can be generated and launched. The SimSO design permits consideration of different time overhead that takes places in system lifetime. This consists of direct overheads like context switches and scheduler calls but also with the simplistic system locks to prevent parallel execution of scheduler. These direct overheads are subjected over the processor that is needed to be occurred. In order to perform the scheduling algorithm implementation, a scheduling simulator "SimSo1" is used for scheduler scripting as well as scheduler execution..

4.1 Scripting a Scheduler

The initial step of the real time scheduling mechanism is to specify the scheduling algorithm so that it can deal with the any form of online schedulers like semi-partitioned, partitioned, global etc., schedulers. The scheduler implementation over the real time system should be realistic where the processor running the scheduler should have performance impact over the schedulability. The finite precision of timer introduces a small difference in the experimental scheduling and theoretical schedule which causes a

major concern. A significant aspect of the SimSO simulator is that it offers simple experimentation. The tool extracts features from python language which has beneficial factors to the scientific industry. Each schedulers in the study composed of <200 lines of codes which makes simple in real time implementation. The simulator loads the inherited python class dynamically over the simulator. The method considered for the simulator implementation is given in Fig.2.

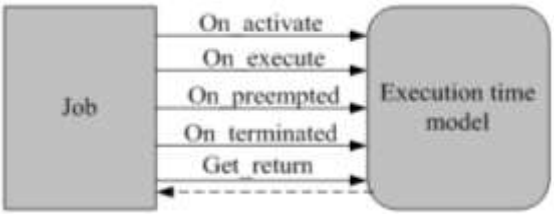

Figure.2. Interfacing of execution model.

- Once the simulation is begin, the "init" method is called to perform the scheduler initialization.
- After activating the job "On_activate" method can be called.
- After job execution/abortion, the "On terminated" method is called.

The scheduling decisions are returned in "Schedule" method and are called when a processor requests to take a scheduling decision. The processor request is mainly performed during a job activation/ termination/by a timer.

4.2 Scheduler Execution

The simulation is considered to analyze the system schedulability where it is assumed that a task meets their respective jobs in worst-case execution time (WCET). But, WCET is very problematic one as all the jobs hardly reach the respective upper bound of execution and it is difficult to meet all the jobs together. However, comparing to the other scheduling schemes the WCET has better performance. The SimSo tool already composed of many execution time models in which simple models are consists of task WCET for their execution time. Other models use random time to meet the given average execution time (AET) during each job. The AET model exhibits normal distribution through standard deviation, mean and bounded by WCET. The rest of the models able to detect the migrations and preemptions and are extended to WCET for job execution in fixed time. In this manuscript latter model is considered where the job execution time is depends on the active events. This model is effective as it performs the simulation of shared cached and the consequences of job executions depending on external aspects.

IV. RESULTS AND DISCUSSION

The analysis of the different scheduling mechanism is conducted by using SimSO where schedulers are programmed. The manuscript works on preemptions and migrations over the function of the tasks, for different processors and load by comparing DP-WRAP, EKG and EDF schedulers.

5.1 Defining Characteristics

The characteristic of the simulated system is defined by using following parameters:

- System utilization: 85%, 95%
- Tasks: 20:10:100
- Processors: 2, 4, 8

For every configuration (utilization, tasks, and processors) of the system, five different task sets are generated by using the methods provided by the SimSo i.e., (9 tasks × 3 Processors × 2 system utilization × 5 × 5) of total 1350 systems.
In the analysis process, a Rand_Fixed_Sum algorithm is used to know the task utilizations while the periods are selected randomly in a log-uniform distribution (2ms to 100ms). The time interval used for WCET time model to simulate each system is 0-1000ms.

5.2 Simulation Results

Once the simulation is done, total job migrations and preemptions are extracted by using results object created by the model object. The preemptions (i.e., scheduler is called but not taken any decision) are not are not considered in analysis. The load on the EKG, DP-WRAP and EDF scheduler are given Table 4.1, 4.2 and 4.3 respectively.

Table 4.1. Load on EKG scheduler

CPU	Total Load	Payload	System load
CPU-1	1.0	1.0	0.0
CPU-2	1.0	1.0	0.0
CPU-3	0.2879	0.2879	0.0
Average	0.7626	0.7626	0.0

Table 4.2. Load on DP-WRAP scheduler

CPU	Total Load	Payload	System load
CPU-1	1.0	1.0	0.0
CPU-2	1.0	1.0	0.0
CPU-3	1.0	1.0	0.0
Average	1.0	1.0	0.0

Table 4.3. Load on EDF scheduler

CPU	Total Load	Payload	System load
CPU-1	0.8530	0.8030	0.0500
CPU-2	0.9000	0.8640	0.0360
CPU-3	0.9250	0.8800	0.0450

CPU-4	0.9060	0.8570	0.0490
Average	0.8960	0.8510	0.0450

From Table 4.1, 4.2 and 4.3, the load on EDF is much different than EKG and DP-WRAP as it meets deadlines. The saved results are tabulated as shown in Table 4.1, 4.2 and 4.3 are converted into charts by using Microsoft Excel. The Fig.5.1, 5.2 and 5.3 shows analysis for three processors and system utilization (95%).

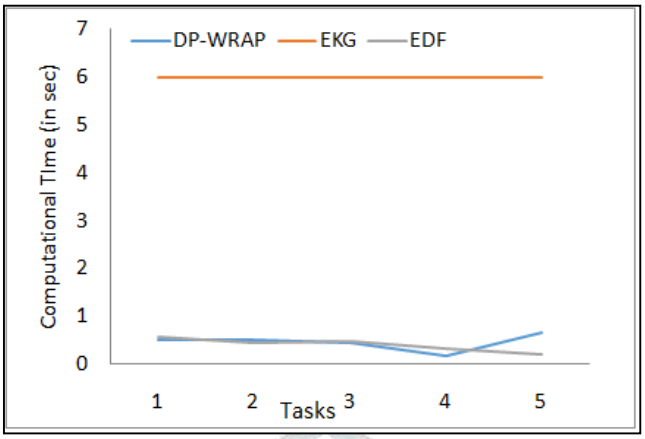

Figure.5.1. Computation time analysis

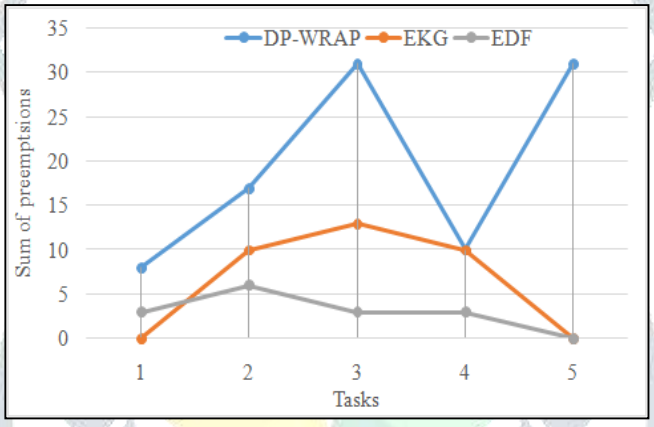

Figure.5.2. Preemption analysis

From the Figure.5.3 it is observed that DP-WRAP generates lot of migrations. The results for EKG scheduler is are getting results with the processors like EDF scheduler. With 5 tasks, EKG and DP-WRAP schedulers works like partitioned scheduler in most of the cases. Most of the time, the schedulers like DP-WRAP and EDF are considered as partitioned scheduler most of the time and also DP-WRAP scheduler generates lot of migrations. Hence, EDF scheduler is considered as more efficient than DP-WRAP and EKG schedulers based on preemptions and migrations. However, some of the jobs are aborted due to deadline misses

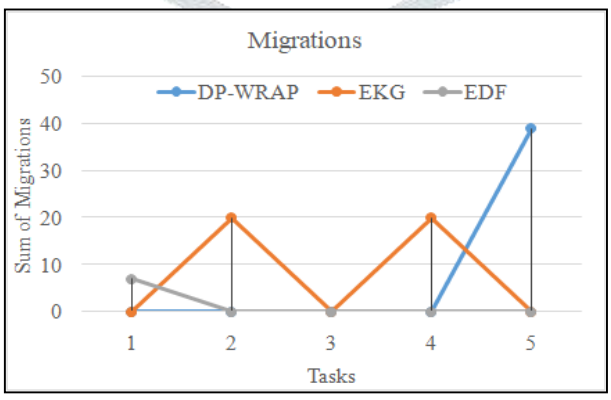

Figure.5.3. Migration analysis

V. CONCLUSION

The tasks in real time scenario are required to be attended immediately where the tasks are meant with controlling of events (or) respond to them. However, handling these tasks is very complicated. Hence, a study is conducted to compare different scheduling algorithms are analyzed and compared their performance in real time scheduling of multiprocessor architectures. The analysis is carried by using SIMSO tool which is open source simulation tool. From the analysis it is observed that: (i) Load on EDF is much different than EKG and DP-WRAP as it meets deadlines, (ii) DP-WRAP generates lot of migrations. The results for EKG scheduler is are getting results with the processors like EDF scheduler and (iii)With 5 tasks, EKG and DP-WRAP schedulers

works as a partitioned scheduler and also DP-WRAP scheduler generates lot of migrations. Hence, EDF scheduler is considered as more efficient than DP-WRAP and EKG schedulers based on preemptions and migrations. However, some of the jobs are aborted due to deadline misses. Hence, EDF offers significant solution for preemptions and migrations. Thus, the proposed approach attained (a) Effective real-time scheduling, (b) Simple easy, fast and flexible simulation (c) Classification of scheduling mechanisms and (d) Comparison of different scheduling policies.

The future scope of the manuscript is that it can be further extended with different set of scheduler techniques at different number of tasks, processor and system utilization. Also, the project can be considered for real time applications like Networked Multimedia Systems, Air Traffic Control Systems, command Control Systems etc.

VI. ACKNOWLEDGMENT

We would like to thanks Dr. Ramesh S, Head of Department of Electronics and Communication Engineering, Dr. Ambedkar Institute of Technology, Bengaluru, India; Also, all faculty members of Department of Electronics and Communication Engineering, for excellent support to carried out this work and giving us inspiration at each and every time. Finally, we would like to thanks our family members who always support, help and guide us during our work.

REFERENCES

[1] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in Proc. of RTCSA, 1998.
[2] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for lowpower embedded operating systems," in Proc. of SOSP '01, 2001.
[3] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, "RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor," in Proc. of RTSS, 2011.
[4] I. Ripoll, A. Crespo, and A. Mok, "Improvement in feasibility testing for real-time tasks," Real-Time Systems, vol. 11, no. 1, 1996.
[5] F. Singhoff, J. Legrand, L. Nana, and L. Marc´e, "Cheddar: A flexible real time scheduling framework," Ada Lett., vol. XXIV, no. 4, 2004.
[6] A. Srinivasan and S. Baruah, "Deadline-based scheduling of periodic task systems on multiprocessors," Inf. Process. Lett., vol. 84, no. 2, 2002.
[7] R. Urunuela, A.-M. D´eplanche, and Y. Trinquet, "STORM a simulation tool for real-time multiprocessor scheduling evaluation," in Proc. of ETFA, 2010.
[8] Chéramy, Maxime, Pierre-Emmanuel Hladik, and Anne-Marie Déplanche. "Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms." 2014.
[9] Chéramy, Maxime, et al. "Simulation of real-time scheduling with various execution time models." Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014). IEEE, 2014.
[10] Devi, U. and Anderson, J. "Tardiness bounds under global edf scheduling on a multiprocessor", In Proc. of the 26th IEEE Real-Time Systems Symposium (RTSS), 2005.
[11] H. Liu, Q. Lin and J. Huang, "Design and analysis of hybrid scheduling algorithm for messages and tasks in networked control systems," Proceedings of the 29th Chinese Control Conference, Beijing, 2010, pp. 4283-4288.
[12] J. Li, G. Zheng, H. Zhang and G. Shi, "Task Scheduling Algorithm for Heterogeneous Real-time Systems Based on Deadline Constraints," 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 2019, pp. 113-116.
[13] Z. Xiaoqing, H. Yajie and A. Chunlin, "Data-Dependent Tasks Re-scheduling Energy Efficient Algorithm," 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China, 2018, pp. 2542-2546.
[14] K. Zhang, B. Qi, Q. Jiang and L. Tang, "Real-time periodic task scheduling considering load-balance in multiprocessor environment," 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, Beijing, 2012, pp. 247-250.
[15] J. Zhang, D. Mei and M. Yang, "A Heterogeneity Based Heuristic Algorithm for Scheduling Out-Tree Task Graphs," 2015 2nd International Conference on Information Science and Control Engineering, Shanghai, 2015, pp. 4-8.
[16] Y. Zhao, L. Chen, Y. Li and W. Tian, "Efficient task scheduling for Many Task Computing with resource attribute selection," in China Communications, vol. 11, no. 12, pp. 125-140, Dec. 2014.
[17] X. Wang, Z. Li and W. M. Wonham, "Optimal Priority-Free Conditionally-Preemptive Real-Time Scheduling of Periodic Tasks Based on DES Supervisory Control," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 7, pp. 1082-1098, July 2017.
[18] C. A. Rincón C, X. Zou and A. M. K. Cheng, "Real-Time Multiprocessor Scheduling Algorithm Based on Information Theory Principles," in IEEE Embedded Systems Letters, vol. 9, no. 4, pp. 93-96, Dec. 2017.
[19] Z. Dong et al., "A General Analysis Framework for Soft Real-Time Tasks," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 6, pp. 1222-1237, 1 June 2019.
[20] J. Sun, H. Cho, A. Easwaran, J. Park and B. Choi, "Flow Network-Based Real-Time Scheduling for Reducing Static Energy Consumption on Multiprocessors," in IEEE Access, vol. 7, pp. 1330-1344, 2019.
[21] Siddesha K, and Jayaramaiah G V, "Heterogeneous Processor Scheduling Using Adaptive Particle Swarm Optimization For DVFS Enabled Embedded Systems," in International Journal of Advanced Science and Technology, Vol 29, Issue 12, pp. 488-499, 2020.
[22] De Martini, Daniele, et al. "Peak load optimization through 2-dimensional packing and multi-processor real-time scheduling." Proceedings of the Computing Frontiers Conference. 2017.
[23] Huang, Kai, et al. "Energy-efficient fault-tolerant mapping and scheduling on heterogeneous multiprocessor real-time systems." IEEE Access 6 (2018): 57614-57630.
[24] Doan, Duy, and Kiyofumi Tanaka. "A Novel Task-to-Processor Assignment Approach for Optimal Multiprocessor Real-Time Scheduling." 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). IEEE, 2018.
[25] Lee, Hyunsung, et al. "Panda: Reinforcement Learning-Based Priority Assignment for Multi-Processor Real-Time Scheduling." IEEE Access 8 (2020): 185570-185583.