RESEARCH ARTICLE OPEN ACCESS

EDHS Schedulability Analysis for Real-Time Multiprocessor Scheduling

Rula Mreisheh [1], Mohammed Hijazieh [2]
PhD Student [1]
Department of Computer and Automatic Control Engineering
Assistant Professor [2]
Department of Computer and Automatic Control Engineering
Tishreen University
Lattakia-Syria

ABSTRACT
Scheduling algorithms play a main role in the design of real-time systems. Due to high processing and low price of multiprocessors, real-time scheduling in such systems is more motivating but more complicated. Earliest Deadline and Highest-priority Split (EDHS) is a scheduling algorithm for sporadic tasks performs similar to the traditional partitioning scheduling, as long as tasks are successfully partitioned, but if a spare capacity of each individual processor is not enough to accept the full execution of the task, then a task is allowed to be shared between multiple processors, beyond partitioning. In this paper, we measure the EDHS number of migrations for sporadic tasks under different utilization bounds. We also compare the number of context switches, average deadline misses and tasks average waiting time of EDHS algorithm with well-known algorithms such as Partition Earliest Deadline First (P-EDF) and Partition Rate Monotonic (P-RM). In this comparison, the number of processors and tasks has been increased to evaluate the effect of this increment on the performance of the aforementioned scheduling algorithms.
Keywords: **-** Multiprocessor System, Real Time Scheduling, EDHS algorithm, Sporadic tasks.

I. INTRODUCTION

The recent and rapid growth of real-time applications increases the use of computers to control safety critical real-time functions over the past few years. As a result, real-time systems [1] where the correctness of the system behaviour depends on both the logical results of the computation and the time at which these results are produced, have become the focus of much study.

Multiprocessor scheduling techniques fall into two general categories [1,2]: Global and Partitioning scheduling algorithms. In the global scheduling scheme, all ready tasks are kept in a global queue which is shared among all processors. In the partitioning scheduling scheme, the tasks are statically partitioned and all tasks in a partition are assigned to the same processor and always executed on it. Tasks are not allowed to migrate, therefore the multiprocessor scheduling problem is transformed to many uniprocessor scheduling problems. Recent studies have made a new class of multiprocessor scheduling, so-called semi-partitioning [3]. In semi-partitioning scheduling, most of tasks are assigned to particular processors, but the rest of tasks are allowed to migrate between processors. As a result, it usually performs better than partitioning, while the number of migrations is much smaller than global scheduling [3].

Earliest Deadline and Highest-priority Split (EDHS) is a semi-partitioning scheduling algorithm presented by Kato & al. which improves schedulable multiprocessor utilization by 10 to 30% over the traditional partitioning approach when it schedules sporadic tasks [4].

For EDHS [4,5], a traditional partitioning is performed before splitting the worst-case execution time (e_i) of a task. If the partitioned scheduling fails, the remaining e_i portions are Shared on two or more processors. Each part of the task is defined in order to fill a processor. Kato & al. chose to assign at most one migrating task to each processor. A task always migrates in the same way, between the same processors and at the same time of their execution. Here, the notion of semi-partitioned scheduling takes its full meaning.

The reminder of this paper is organized as follows: In section II, we summarize the scheduling criteria we have considered in this paper. In section III, we introduce previous works that study and evaluate real time scheduling algorithms. In section IV, we describe our system model that we have carried out to evaluate scheduling algorithms. We analyze the performance of the EDHS algorithm depending on different values of parameters and we compare it with well-known algorithms such as Partition Earliest Deadline First (P-EDF) and Partition Rate Monotonic (P-RM) in section V. Finally, a conclusion is presented in section VI.

II. SCHEDULING CRITERIA

Many criteria have been suggested for comparing real time scheduling algorithms. Those characteristics are used to compare and to make an essential difference in which algorithm is judged to be the best. The criteria we have considered in this paper include the following:

1. *Migrations*: we say that a task migrates if it is moved from one processor to another during its lifetime. If tasks can change processor only at job boundaries, we say that task migration is allowed; we call instead job migration, the possibility of moving a task from a processor to another during the execution of a job [6].
2. *Waiting time*: [7] is the total time a task has been waiting in ready queue,
3. *Context switches*: [7] is a task of storing and restoring context (state) of a preempted task, so that execution can be resumed from the same point at a later time. Context switch [8] makes multitasking possible. At the same time, it causes unavoidable system overhead. The cost of context switch may come from several aspects. The processor registers need to be saved and restored, the operating system kernel code (scheduler) must execute, the translation look-aside buffer (TLB) entries need to be reloaded, and processor pipeline must be flushed. These costs [8] are directly associated with almost every context switch in a multitasking system.
4. *Deadline misses:* in a real-time system if a task cannot complete its execution and misses its deadline for any reason, it not only wastes the CPU time but also minimizes the chance of the other tasks waiting for execution to be completed.

So, a good scheduling algorithm should possess the following characteristics:

- Minimize task migrations
- Minimize task average waiting time
- Minimize context switches
- Minimize deadline misses

III. RELATED WORKS

Liu and Layland [9] came up with the static scheduling algorithm Rate Monotonic (RM) of real-time operating systems which is first scheduling algorithm implemented in almost all the real time systems. Processor utilization can be increased by using dynamic scheduling algorithms, such as the Earliest Deadline First (EDF) [9] or the Least Slack algorithm [10]. Both algorithms have been shown to be optimal and achieve full processor utilization, although EDF can run with smaller overhead. Authors in [11] have made an extensive study on memory management and scheduling in real-time systems. The framework for evaluation of real time systems has been described in [12], and this article makes a good point of analysing and comparing real-time operating system under different load conditions. Authors in [13] have made a worst case response time analysis of real time tasks under hierarchical fixed priority pre-emptive scheduling and they have developed an modified Round Robin algorithm for scheduling in real time systems. Kato & al. [4] came up with (EDHS) which is a semi-partitioning scheduling algorithm and they chose the success ratio as the key factor to evaluate its performance in the case of first-fit, best-fit and worst-fit. It has been proved that EDHS improves schedulable utilization by about 10% over P-EDF and this evaluation has been done by using only 16 processors [4]. There are some other works which studied and evaluated semi-partitioned scheduling based on the earliest deadline first algorithm and other algorithms [5,14].

However, to the best of our knowledge, the number of tasks migrations, context switches, and average deadline misses and tasks average waiting time of EDHS has not been measured. In this paper, we measure the EDHS number of migrations for sporadic tasks under different utilization bounds. We also compare the number of context switches, average deadline misses and tasks average waiting time of EDHS algorithm with well-known algorithms such as Partition Earliest Deadline First (P-EDF) and Partition Rate Monotonic (P-RM). In this comparison, the number of processors and tasks has been increased to evaluate the effect of this increment on the performance of the aforementioned scheduling algorithms.

IV. SYSTEM MODEL

This paper considers the scheduling of n in-depended sporadic tasks with implicit deadlines (the deadline of the task is equal to its period) on a platform of m identical multiprocessor. Two parameters are used to describe a task T_i; its worst-case execution time e_i as well as its period p_i. The period of sporadic task [1,15] is a minimum inter-arrival time, that is, a minimum interval of time between two successive activations, because a sporadic task is activated irregularly with this bounded rate.

The time constraints of task T_i is usually a deadline D_i. An instance of a task (i.e., release) is known as a job and is denoted as $T_{ij}= (e_{ij}, p_{ij})$ where j=1, 2, 3,…, and e_{ij} denotes the worst-case execution requirement of job T_{ij} where p_{ij} denotes its period. The deadline of a job is the arrival time of its successor. For example the deadline of the job of Tij, would be the arrival time of job $T_{i(j+1)}$, that is at $(j + 1)\, p_i$. The laxity of a job T_{ij} at time t, denoted $l_{ij\,,t}$, is the time that T_{ij} can remain idle before its execution should be started, i.e. $l_{ij\,,t}= p_{ij} - e_{ij\,,t} - t$, where $e_{ij\,,t}$ denotes the remaining execution of job T_{ij} at time t.

One more important parameter that is used to describe a task Ti is its utilization [1,2,5,16] and is denoted as:

$$u_i = e_i / p_i \qquad (1)$$

The utilization of a task is the portion of time that it needs to execute after it has been released and before it reached its deadline. U_{sum} denotes [1,2,5,16] the total utilization of a

given task set T, whereas U_{max} describes its maximum utilization.

$$U_{sum} = \sum u_i \quad (2)$$

A task set T is said to be schedulable on m identical multiprocessor [16] if and only if:

$$U_{sum}(T) <= m \;\&\&\; U_{max}(T) \leq 1 \quad (3)$$

The concerned algorithms in this paper are:

1. Earliest Deadline and Highest-priority Split (EDHS)
2. Partition Earliest Deadline First (P-EDF)
3. Partition Rate Monotonic (P-RM)

V. EXPERMENTAL RESULTS

In the experiments, the values of the parameters are considered as below, unless mentioned otherwise.

1. Tasks are preempt-able.
2. The period of tasks is a random number with a uniform distribution between 1 and 100.
3. The number of tasks is 4, 8, 16, 32, 64 and 128 which are executed on 2, 4, 8, 16, 32 and 64 processors respectively, i.e., the number of tasks is double the number of processors (n=2m).
4. We have generated 1000 task sets with full utilization $U_{sum}(T) = m$.
5. The results have been obtained in an observation window between 1 and 10000.

A. *Experiment 1: The task migrations of EDHS*

In this paper, we measure the number of migrations of the EDHS algorithm for sporadic tasks under different utilization bounds and the results we have obtained are shown in Fig. 1.

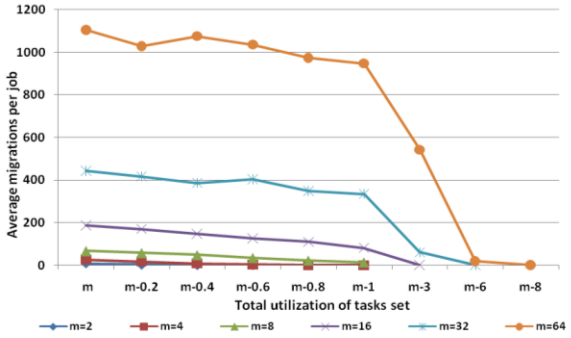

Fig. 1 The average migrations of the EDHS algorithm under different utilization bounds.

Since a migration occurs when the semi-partitioned technique is used, the results show no migration with low task sets utilization. Our graph focus on the range of utilization [m;m-8].

B. *Experiment 2: Tasks average waiting time of EDHS, P-EDF and P-RM*

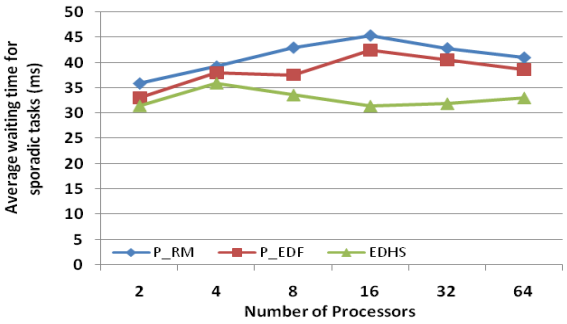

Fig. 2 Comparison of average waiting time between EDHS algorithm and the best known algorithms.

Fig. 2 shows the results of simulations based on tasks average waiting time. We have carried out a study to compare the tasks average waiting time of P_RM, P_EDF and EDHS algorithms. Since a good scheduling algorithm should minimize the waiting time, the results show that the EDHS algorithm outperform the other algorithms based on partitioned scheduling that because EDHS algorithm allows only the un-partitioned tasks to migrate between processors thus it minimizes the waiting time of tasks in the ready queues.

C. *Experiment 3: The context switches of EDHS, P-EDF and P-RM*

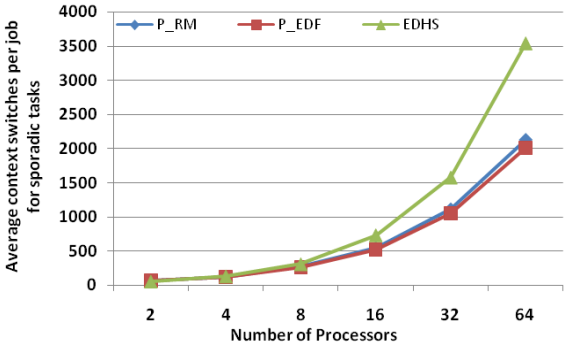

Fig. 3 Comparison of average context switches per job between EDHS algorithm and the best known algorithms.

The average number of context switches of EDHS, P_EDF and P_RM is shown in Fig. 3. As the number of processors increases, the number of context switches of all the algorithms increases. In this case, the partitioned policies show a better performance.

D. *Experiment 4: Average deadline misses of EDHS, P-EDF and P-RM*

In this experiment we compare the average number of' deadline misses of EDHS, P_EDF and P_RM algorithms and the results are shown in Fig. 4.

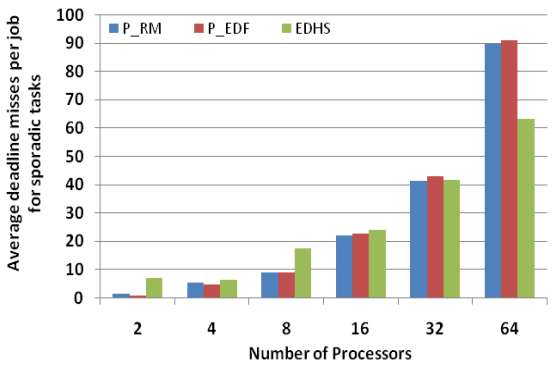

Fig. 4 Comparison of average deadline misses per job between EDHS algorithm and the best known algorithms.

According to our analysis, the results show that by increasing the number of processors to 64, EDHS algorithm lessens the average deadline misses about 22% more than partitioned policies. Since EDHS algorithm receives one un-partitioned task on each processor, the large number of processors allows to receive large number of shared tasks and thus most of shared tasks are able to meet their deadlines.

VI. CONCLUSION

From the EDHS comparative study with partitioned policies (1000 random sporadic task sets have been generated with full utilization U_{sum} (T) = m), it can be concluded that even if EDHS algorithm minimizes deadline misses and tasks waiting time in queues, it causes large number of context switches. Since the concept of time constrain is of such importance in real-time application systems, EDHS scheduling algorithm can meet the needs of real-time applications, in soft applications, it is still needed to be improved to meet the evolving needs of critical real-time systems.

REFERENCES

[1] S. Baruah and J. Goossens, "Scheduling Real-time Tasks: Algorithms and Complexity," *Computer and Information Science Series*, vol. 28, p. 38, 2004.

[2] A. Mohammadi and S. G. Akl, "Scheduling Algorithms for Real-Time Systems," Queen's University, Canada K7L 3N6, Tech. Rep.2005-499, 2005.

[3] B. Andersson and K. Bletsas, "Sporadic Multiprocessor Scheduling with Few Preemptions," in *Proc. of the Euromicro Conference on Real-Time Systems*, 2008, pp. 243–252.

[4] S. Kato and N. Yamasaki, "Semi-partitioning technique for multiprocessor real-time scheduling," *in Proc. of the WIP Session of the 29th Real-Time Systems Symposium (RTSS)*, IEEE Computer Society, 2008, p. 4.

[5] L. George, P. Courbin, and Y. Sorel, "Job vs Portioned Partitioning for the Earliest Deadline First Semi-Partitioned Scheduling," *Journal of Systems Architecture*, vol. 57, pp. 518–535, May. 2011.

[6] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 553 – 566, June, 2008.

[7] A. Abdulrahim, S. E. Abdullahi, and J. B. Sahalu, "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm," *International Journal of Computer Applications*, vol. 90, no. 4, pp. 27-33, Mar. 2014.

[8] C. Li, C. Ding, and K. Shen, "Quantifying The Cost of Context Switch," San Diego, ExpCS, June. 13-14, 2007.

[9] C. Liu and J. Leyland, "Scheduling algorithm for multiprogramming in a hard real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, pp. 46-61, Jan. 1973.

[10] A. K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment, " Ph.D. thesis, MIT, 1983.

[11] S. Baskiyar and N. Meghanathan, "A Survey On Real Time Systems," Informatica (29), pp. 233-240, 2005.

[12] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behaviour," *IEEE Real-Time Systems Symposium*, 1989, p. 5.

[13] C. Yaashuwanth and R. Ramesh, "Design of real time scheduler simulator and development of modified round robin architecture," *International Journal of Computer Applications*, vol. 10, no. 3, pp.43-47, Mar. 2010.

[14] M. K. Bhatti, C. Belleudy, and M. Auguin, "A semi-partitioned real-time scheduling approach for periodic task systems on multicore platforms," *in Proc of the 27th Annual ACM Symposium on Applied Computing, Riva, Trento, Italy*, 2012, pp. 1594-1601.

[15] V. B. Alberto and M. Spaccamela, "Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems," *in Proc. of the 18th European Symposium on Algorithms*, 2010, pp. 230-241.

[16] B. B. Brandenburg, "Scheduling and Locking in Multiprocessor Real-Time Operating Systems," Ph.D. thesis, the Department of Computer Science, North Carolina, 2011.