IJCSN International Journal of Computer Science and Network, Volume 3, Issue 5, October 2014
ISSN (Online) : 2277-5420 **www.IJCSN.org**
Impact Factor: 0.274

A Comparison of Global EDF and LLREF Scheduling Algorithms

[1] Valma Prifti, [2] Bledar Balla, [3] Romina Zane, [4] Juliaj Fejzaj, [5] Igli Tafa

[1] Polytechnic University of Tirana, Faculty of Economy Engineering

[2] Polytechnic University of Tirana, Information Technology Faculty

[4] Tirana University, Information Faculty

[5] Polytechnic University of Tirana, Information Technology Faculty

Abstract - In this paper we are going to analyze and trying to compare EDF and LLREF scheduling algorithms, used in multiprocessor platforms. We will briefly describe classic EDF, global EDF, which is an extension of EDF, and the newer LLREF algorithm. Then we will analyze the ability of global EDF in scheduling task sets in random manner and compare with the LLREF one, which can schedule any task set when the total utilization is smaller then number of processors. We will examine optimally and the overhead of these two algorithms. Analyzing three parameters such as: schedulability, task migration and scheduler invocations, we will try to be as specific as possible with our conclusions.

Keywords - *EDF, LLREF, algorithm, scheduling, simulator, deadline, jobs, task, invocations, processor, schedulability, execution, utilization.*

1. Introduction

In real time systems, basic units of work (jobs) must be executed in a timely manner. In most real time platforms every single job is characterized by three parameters: an arrival time, an execution requirement and a deadline with the condition that the job must be executed in an interval between the arrival time and the deadline. In uniprocessors, where all the job have to execute on one single shared processor, one of the most optimal scheduling algorithm is EDF (Earliest Deadline First).A scheduling algorithm is called optimal if it fulfills almost all part of requirements that a certain job may have. Scheduling algorithms, such as EDF which are mostly used for uniprocessors, may not be used when apply in multiple processors. So, what happens in multiprocessor platforms? Is EDF still an optimal scheduling algorithm? We will try to explain EDF and global EDF optimally, obviously given the adequate motives about that. In multiprocessor systems is used the extension of EDF, called global EDF. We will compare global EDF with LLREF which is based on the principal that all tasks meet their deadlines when the total utilization demand is smaller than the utilization capacity of the platform. It is still difficult nowadays to schedule in multiprocessor systems, and this remains a challenge in modern operating systems. The first scheduling algorithm we are going to discuss, is global EDF and the second is LLREF. We implemented these algorithms in an open source simulator, called RTSIM (6). In the end of this paper we will graphically present our simulation results.

2. Scheduling Algorithms

2.1 Global EDF

EDF (Earliest Deadline First) is a widely-used dynamic scheduling algorithm (1). It is used in preemptive uniprocessors. In this algorithm, the priority of each instance of a task is determined by its deadline. The highest priority is given to the task with the earliest deadline. The utilization bound of this classic algorithm is 100%, because of the equality of deadline and period of the processes that are scheduled. The schedulability of EDF is :

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

Where {Ci} are worst case computation times and "n" is the total number of processes, {Ti} is the absolute period which is assumed to be equal to the relative deadlines.

If there is any active task, the processor will execute it rather than being idle, so edf is called work-conserving.

When the system is overloaded, there is a dysfunction of EDF and this is a considerable disadvantage for a real time system designer. Meanwhile the tasks that are chosen to schedule, miss their deadlines, other tasks that are waiting for their turn, risk to miss their deadline too, so EDF shows a really poor performance. It is also difficult for these algorithms to implement in hardware, because of the difference in the deadlines range. In EDF scheduling it is possible to deal with other types of processes and to use servers for overloads.

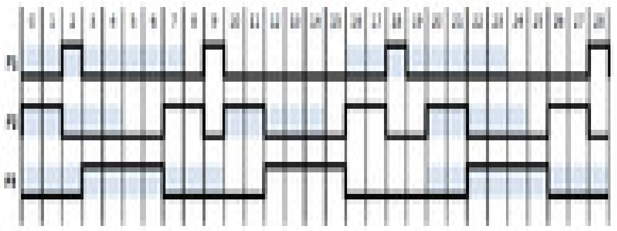

Fig.1 Timing diagram showing part of an Edf schedule

The utilization is:

$$\left(\frac{1}{8}+\frac{2}{5}+\frac{4}{10}\right)=\left(\frac{37}{40}\right)=0.925=\mathbf{92.5\%} \qquad (2)$$

As we mentioned above, EDF is not an optimal scheduling in multiprocessor platforms. It might miss deadlines even if processors almost half of the time,are iddle. The high overhead and the low possibility to apply to different models for jobs and processors, make some optimal multiprocessor scheduling less desirable than others. To improve the performance of scheduling, in order to use the EDF, is used gEdf (global EDF) which is, as we mentioned, an improvement of the classic one. Global EDf is one of the first dynamic priority-driven scheduling algorithms that are proposed. In gEDF jobs are ordered by their earliest absolute deadline. The highest priority processes are those who are choosen by processors to execute.

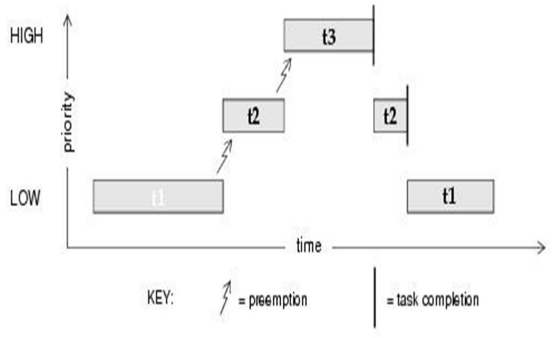

Fig 2 Tasks dependence from preemption

When the new job arrives or when an actual job finishes, occurs the scheduling process. Two are the reasons for a migration to occur: the first is the number of processors and the second is when preemption occurs. It is logical that the likehood of migration is as high as the possibility of preemption to happen.

After an actual job is finished, the scheduler chooses to exexute the no-active highest priority job. It does not depend on which task was the processor execute before. This is known as idle-active situation because of the idle-active transitions that occurs between the scheduler and processor. If this situation lasts in time for an unknown period, a job is forced to migrate between CPUs. We mentioned also that the other main reason of migrating is the number of CPUs. The only reason a preempted task might not be forced to migrate in CPUs is if this task finishes before the other active job. We are allowed to discuss this only in multiprocessor systems, in an uniprocessor system is logical that migration can not happen. For P=2 preempetive jobs have a migration probabillity of 50%, for P=10 the percentual of migration increase in 90%. As CPU counts increase, the percentual of migration also increases. Based on schedulability bound, we can say that if total utilization is less than 1, it is schedulable, otherwise it is not. So, gEDF can correctly schedule any tasc system, the Umax of witch, is at most *m/(2m-1)* , as long as Usum, the total utilization of all tasks is not more than *m2 /(2m-1).* If the utilization is greater than P, the schedule do not occurs. The following formula is used to calculate the bound:

$$\sum \frac{c_i}{d_i} \leq max\left\{\frac{c_i}{d_i}\right\}(P-1) \qquad (3)$$

To illustrate this, take a simple example: P=4, Ti(1,2) i ϵ [1,5]. The condition evaluates in $\frac{5}{2} \leq \frac{5}{2}$. If any task were longer than this, they might not fit in process. In the figure below are shown five identical tasks.

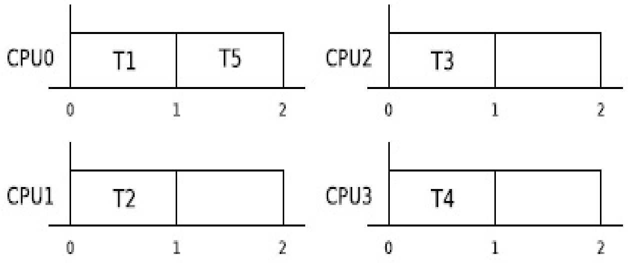

Fig 3 Global EDF schedule of five identical tasks with utilization 1/2

2.2 LLREF

LLREF (Largest Local Remaining Execution First) is an optimal algorithm which is used to schedule periodic task sets in identical multiprocessors. LLREF is based on the princip that all tasks meet their deadlines when the total utilization demand is smaller or equal with the utilization capacity of the platform. The deadline is supposed to be equal with the period. Task migrations and preemptions are supposed to be negligible but they are fully allowed as long as the same task is not executed parallelly on more than one processor. A task set τ can be scheduled only if it satisfies the conditions below:

$$U(\tau) \leq m, \text{ and } u_{\max}(\tau) \leq 1, \qquad (4)$$

where the $u_{\max}(\tau)$ is the total utilization of all tasks in τ. As long as a task set τ satisfies two conditions above (the local utilization is minor than the total utilization capacity), LLREF schedules the all the tasks. LLREF algorithm is based on an abstraction which is known as Time and Local Plane (3). This abstraction determines when a task must be scheduled in order to meet its deadline. A visual idea of this concept, is given in the figure below.

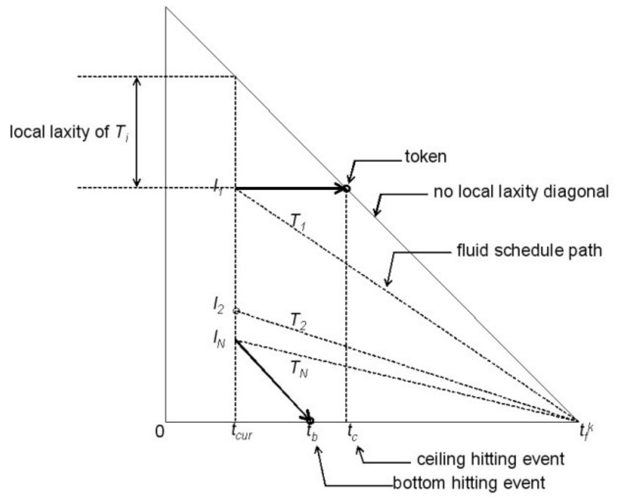

Fig 4 Scheduling in T-L plane

As the figure illustrates, two main elements of this abstraction are: a) Time (real time) and b) the time that remains from total execution time of a given task . The two axis are the same length and the slope of the no local laxity diagonal is -1. The tokens above, represent a task situation in a certain time. The remaining execution time of a task is called local execution time because it shows the time that must be consumed until the end of the period. The decisions of the scheduler change by the time,as result tasks tokens move in the T-L plane. All the tokens should be between the "no local laxity diagonal" and the horizontal line. When a task is choosen by scheduler to execute the token moves diagonally down, otherwise it moves horizontally. In a multiprocessor system, with *m* processors, *m* tokens are most likely to move diagonally together.The system choose to schedule a task when the previous one has zero remaining execution time. The tasks that have zero local remaining execution time are invalid, so they could not be choosen anymore. Based on the local remaining execution time, there are two task's categories: *active* and *inactive*. LLREF scheduling is a priority scheduling, so,bigger the remaining time is, bigger is the possibility to be selected. If x_t – is number of tasks ready to be choosen from the system for executing, *t*-time interval of executing and *m*- the maximum number of tasks allowed to execute, then the number of tasks with remaining work is:

$$x_t = \min\{m, |\text{Active}(t)|\}. \qquad (5)$$

If a token tends to hit the horizontal line, it means that the task is already executed as long as necessary, so it is turn of another task to select instead. It is called bottom-hitting event.When a task has zero remaining local laxity, the token hits the diagonal line(NLLD) which means that the task needs to be selected immediately to meet the deadline. It is called celling hitting event. When more than *m* tokens hit NLLD (no local laxity diagonal) a *critical moment* is created. After this moment, the system select only few of these tokens. The selection of tokens is based on the princip of Largest Local Remaining Execution First. The tokens that have not been selected move out of the triangle, so they have no possibility to cause a bottom-hitting event, hitting the horizontal line. A critical moment is shown in the figure below.

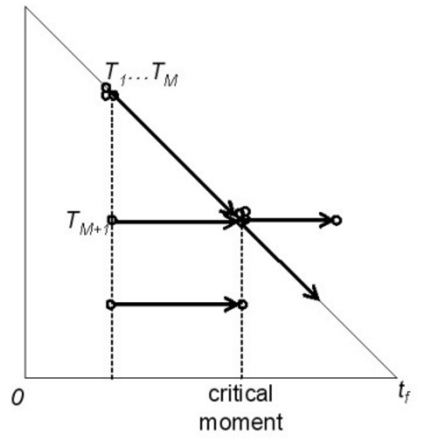

Fig 5 Critical moment

3. Experimental Phase

In order to compare global EDF and LLREF, we implemented them using RTSIM framework, which is an open source library. It is a collection of C++ libraries and is used to simulate real time scheduling algorithms. A basic component of RTSIM is MetaSim which is a generic library for simulating scheduling algorithms and real-time tasks. The processors are managed by a kernel and are allocated to tasks by a scheduler. In our experiment is used a variant of kernel abstraction for multiple processors. For two our algorithms,we also implemented derivations of the scheduler abstraction. We have done some modifications to RTSIM in order to implement the LLREF algorithm.

We have compared three parameters generated from the schedules, of our algorithms. The main purpose of the paper was to compare the overhead of global EDF and LLREf scheduling algorithms and we have done that by comparing schedulability, task migrations and scheduler invocations. Schedulability is determined like the ability of the algorithm to feasibly schedule a given set of tasks. We had to generated task sets with an utilization less than the number of the processors, in order to measure the schedulability of the algorithms. As we knew that global EDF had no guarantee that would be able to schedule these task sets, we recorded all the situations when a task set was not scheduled by EDF. The measurement of schedulability for EDF, shows us that global EDF is a problematic, non-optimal algorithm.

Table 1. Global EDF feasibility statistics

CPUs	Worst miss	μ	σ	Misses
2	1.28	1.82	.161	15%
4	2.27	3.44	.352	14%
8	7.01	7.67	.232	96%

Above, is shown a statistical report of the feasibily of EDF. *Worst miss* is the lowest nivel of utilization that missed. Third and fourth columns show standart deviation that caused *misses.* The number of misses changes with the number of CPUs: the greater the number of CPUs, the greater the the number of misses. Since LLREF is optimal, it was always able to schedule all the taks.

A task migration, in our experiment, occurs when a task is descheduled from one processor and then is rescheduled on another processor. Scheduler invocations for EDF and LLREF occur at different times. The scheduler for global EDF, was invoked on every arrival and every completition of a task instance. So, the decisions of Global EDF are made on the next task subset and the scheduling is based on the absolute deadlines.

4. Results

4.1 Schedulability

LLREF is more complex than global EDF. Eventhough the global Edf is simpler, it is not optimal because of the losses of guarantees. To avoid that, we increased the number of CPU counts. We noticed that it was an effective way to increase the utilization level where deadline was missed, but it doesn't guarantee her eficence all the time, at all.

4.2 Scheduler Invocations

Since the scheduler invocations are source of overhead, the number of them should be as little as possible. Every tasks set has a running time, within the scheduler must be invoked and the set of tasks must be run. The higher the number of scheduler invocations is, the lower is the remaining run time of tasks.

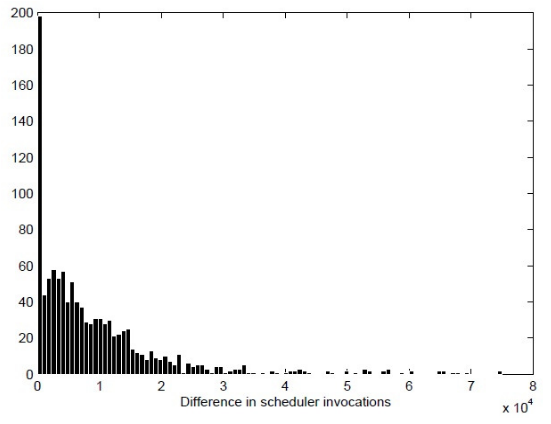

Fig 6 The difference in number of scheduler invocations between EDF and LLREF on 4 processors

The results shows that EDF has fewer invocations than LLREF. In global EDF the invocations occur only when a task arrives or completes. On the other hand, task arrivals in LLREF are considered as primary events. All task sets which are not scheduled by global EDF, are easily feasible using LLREF algorithm. As shown in the figure above, the possibility of having the same number of invocasions in EDF and LLREF, increases with the number of processors.

4.3 Task Migrations

Based on the scheduling algorithm that is used, every task set requires a specific number of task migrations. A main goal of a multiprocessor scheduling algorithm is to minimize the number of task migrations, that is necessary for a task set to run. A task migration requires all resources to be available to the new processor. Our experimental results are shown in the figure below.

When the difference between two our algorithms is negative, this means that LLREF required fewer task migrations than EDF. When the difference is positive, LLREF required more task migration than EDF did.

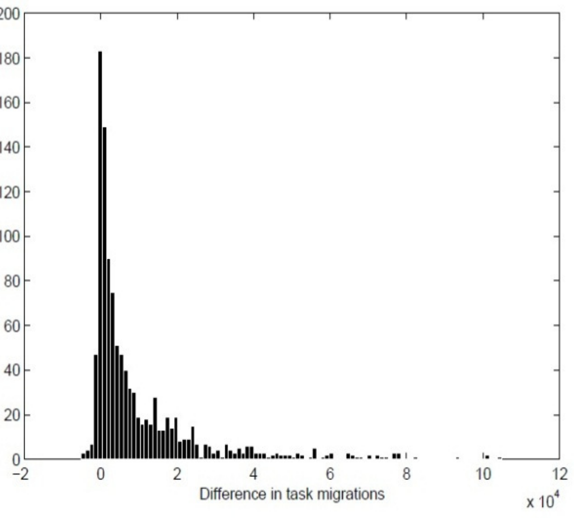

Fig 7 The difference in number of task migrations between EDF and LLREF in 4 processors

During our experiment, we noticed that, for a considerable number of schedules, LLREF and global EDF required a similar amount of task migrations.

5. Conclusion

In this paper we examined and analyzed two scheduling algorithms for uniform multiprocessor systems: global EDF and LLREF. We analyzed three parameters of these algorithms such as: schedulability, task migrations and scheduling invocations. Based on them, we can say that even though EDF is clear and simple, it is a non-optimal scheduling algorithm. Some task sets could not be run using EDF scheduling algorithm. However, the results shows that EDF's overhead, in terms of scheduler invocations and task migrations, is lower than the LLREF one. On the other hand, LLREF is an optimal scheduling algorithm, with a higher overhead than global EDF, but with more task sets run in it. In the end, it's necessary for us to decide which parameter of scheduler is more important: optimally or low overhead, before we choose a scheduling algorithm for multiprocessor systems.

References

[1] Global EDF-based Scheduling with Laxity-driven Priority Promotion Shinpei Kato, Nobuyuki Yamasaki, Department of Computer Science, The University of Tokyo. Journal of Systems Architecture: the EUROMICRO Journal, Volume57 http://dl.acm.org/citation.cfm?id=1975383.

[2] U-LLREF: An Optimal Scheduling Algorithm for Uniform Multiprocessors Shelby H. Funk, Archana Meka, Computer Science Department, Georgia's University, 30606,USA. www.cs.uga.edu/~shelby/pubs/mapsp-shf.pdf.

[3] An Optimal Multiprocessor Algorithm for Sporadic Task Sets with Unconstrained Deadlines Sh.Funk, University of Georgia, Athens, GA, USA. Published in Journal Real Time Systems; Volume 46 Issue 3, December 2010 http://dl.acm.org/citation.cfm?id=1891605.

[4] A Comparison of Scheduling Algorithms for Multiprocessors
Dan McNulty, Lena Olson, Markus Peloquin, December 13, 2010 http://pages.cs.wisc.edu/~markus/750/smp_scheduling.pdf

[5] Energy-aware Scheduling on Multiprocessor Platforms By Dawei Li, Jie Wu http://books.google.al/books.

[6] Retis Lab, Scuola Superiore Sant'Anna. The RTSIM project. http://rtsim.sssup.it.

[7] A Comparison of Global and Partitioned EDF Schedulability Tests for Multiprocessors T. Baker In International Conf. on Real-Time and Network Systems,2005. www.cs.fsu.edu/research/reports/tr-051101.pdf.