EDF scheduling for real time tasks on Multi core

Amit Chakma
Dept. Electrical Engineering
Hamm-Lippstadt University of Applied Sciences
Lippstadt , Germany
amit.chakma@stud.hshl.de

***Abstract*—Earliest Deadline First is one of the best algorithms for a uni-core system ensuring 100 percent CPU usage with guaranteed execution of all tasks. In this paper we will see how the Earliest Deadline First algorithm works on multi-core systems and why it is not a good choice on a multi-core environment[2]. Scheduling on a multi-core system is usually a complex task. We will compare some multiprocessor scheduling algorithms and check their task execution approaches.**

I. Introduction

Technology has been massively improved within the last few decades and single core also got extinct within the time frame. Nowadays processors have dual or more cores for higher performance and at the same time task scheduling also became a very important part for the multi-core processors. We came across multiple algorithms to optimize the workload on each processor.When more than one processor is used, scheduling algorithms face new challenges. The key issue is determining which processor should perform a job, whether it is possible to migrate a task to another processor and when the scheduler should migrate. Task migration generates overhead on bus-systems and caches.

II. EDF on Uniform processor

Earliest deadline first is based on a dynamic scheduling rule that executes the process depending on their deadline. Dynamic scheduling is a scheduling algorithm in which the priorities are made during the execution of the system. The processes with the earliest deadline gets the higher priority. The algorithm executes in preemptive mode. In preemptive mode , the process execution can be halted for other active processes so they can meet the deadline. In a hard real-time system this scheduling helps to execute the tasks with a fixed time eg. air-bag control inside a car. EDF does not predict any assumption for the tasks, so the algorithm can be used for both periodic and aperiodic tasks.

The algorithm executes in preemptive mode on a uni-processor. In uni processor one shared processor is available and all tasks in the system require to execute using a single processor. In preemptive mode , the process execution can be halted for other active processes so they can meet the deadline. Preemptive systems are generally able to allow higher CPU utilization compared to non-preemptive . In a collection of independent jobs , each task is defined by arrival time , an execution requirement and a deadline. In an algorithm, a group of independent tasks can be scheduled by checking their arrival time, the execution requirement and the deadline. EDF will schedule these collections of tasks so they all complete by their deadline. In periodic scheduling where tasks have deadlines equal to their periods , EDF showed a 100% CPU utilization. A set of periodic tasks is schedulable under EDF if the following conditions fullfilled

- all tasks in the task set are independent.
- deadline is equal to period i.e. $D_i = T_i$ for all tasks
- the task set is synchronous.
- and Utilization:

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

where C_i are the worst-case computation-times of the n processes . T_i are their respective inter-arrival periods and assumed to be equal to the relative deadlines[1].

	Execution Time (C)	Deadline (D)	Period (T)
P1	3	7	20
P2	2	4	5
P3	2	8	10

Fig. 1. example for EDF [8]

1) First, we have to check whether the processes are schedulable by calculating the utilization:
   U = C1/P1 + C2/P2 + C3/P3 = (3/20) + (2/5) + (2/10) = 0.75 = 75%
   U = $75 < 100\%$, which shows that the tasks are schedulable.
2) We take the Least common multiple of the periods of the processes to know how many units we need to execute the three processes:
   LCM(20, 5, 10) = 20
   We need 20 units at most to execute the processes.

3) Now we can see the period of P1 = 10, we give it 20/20 = 1 time slices, each is 20 units long and with P2 we get 20/5 = 4 time slices, each with a length of 5 units. Similarly P3 will be given 20/10 = 2 time slice with a length of 10 units. The dividers are marked in green and

the deadlines are marked in black shown in the figure below:

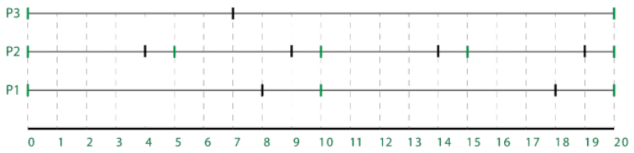

Fig. 2. Black shows the deadline and green for availability of the processes [8]

4) Initially at time unit 0, all processes are ready to execute, but P2 has the nearest deadline compared to P1, P3 so, first P1 is assigned for 2 unit
Now at time 2, only P1 and P3 are ready processes and P3 has the nearest deadline compared to P2, so here P3 has been assigned for 3 units.
Now at time 5, only P1 and P2 are ready processes and P1 has the nearest deadline compared to P2, so here P1 has been assigned for 2 units. At time unit 7 the only ready process is P2. So, we assign P3 for 2 units and remain ideal for 1time unit.
Now at time 10, only P1 and P2 are ready processes and P2 has the nearest deadline compared to P1, so here P2 has been assigned for 2 units and then we assign P1 for 2 units. After 1 unit we execute the available P2.

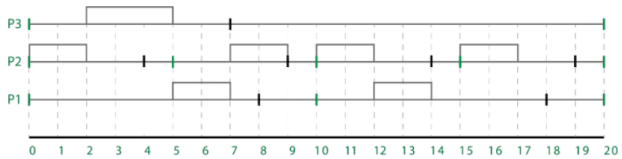

Fig. 3. The diagram shows the scheduling on Periodic EDF [8]

A. Implementation

According to figure 4 which is implemented in Uppaal. From the idle state, tasks arrived at the ready state, the system gives priority to the tasks with the earliest deadline. The tasks with the earliest deadline then execute and the system is complete and reset to start again. This is an abstract explanation of how EDF works.

III. Taxonomy of Multiprocessor Scheduling Algorithms

In Multiprocessor scheduling, allocation problem and priority problem are the main subject areas that scheduling try to optimize. Allocation problems are defined as the processor's decision on which tasks to execute. Priority problems are the problems when and in which order each task should execute with respect to other tasks. There are several types of multiprocessor scheduling algorithms. We can have 3 categories according to the available degree of inter-processor migration and 3 also categories according to when the priority is being changed[3].

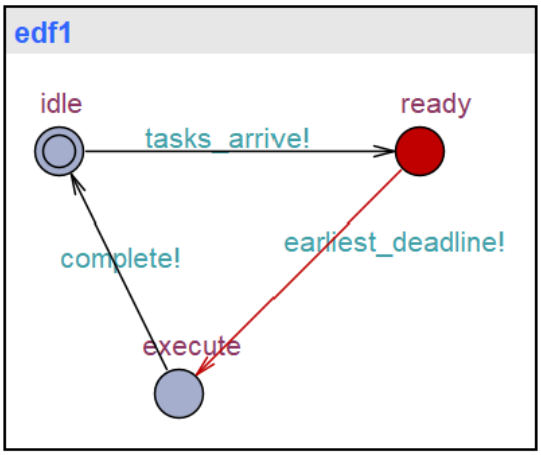

Fig. 4. Uppaal implementation

A. Allocation

Inter-processor migration is generally restricted in real-time systems. In many systems migration from one processor to another processor is prohibited. In real time system theory , there are no such rules that can be followed for migration between processors. Scheduling algorithms fall into one of the following three different categories according to the degree of migration [4].

- No Migration - Every task can be scheduled and executed. No migration is allowed among the processors. This technique can be said to be partitioned. A subset of tasks is assigned to a processor and needs to be scheduled only on a particular processor.
- Task Level Migration - tasks may execute on different processors but each task must execute fully on a single processor. Task level migration is also known as restricted migration
- Job Level Migration - Inter-processor migration is not restricted. A task can be migrated and relocated from one processor to another but parallel execution is limited, so a job cannot be scheduled on more than one processor at the same time. Task level migration is also referred to as full migration.

B. Priority

A scheduling algorithm falls into any of the following three different categories according to the complexity of the priority scheme [4].

- Fixed Task Priority - A unique fixed priority is assigned to each task and this priority will be applied to all of its jobs. Fixed job priority is also referred to as static priority.
- Fixed Job Priority - The different jobs of a task may have different priorities but each job will have the same static priority .

- Dynamic Priority - As the name suggests, a job may have different priority at different points of time. There is no restriction on priorities that can be assigned to jobs.

IV. Multiprocessor Scheduling Approaches

A. Partitioned EDF

In partitioned scheduling tasks are partitioned among the available processors. In other words , tasks are scheduled on a specific processor and executed by that processor only. It can be observed that the multiprocessor behaves like a collection of independent uniprocessor systems . A uniprocessor scheduling algorithm will run for each core to schedule the tasks assigned to that processor. In partitioned scheduling tasks cannot migrate from one processor to another, this activity is extremely prohibited on partitioned scheduling. Some advantages are

- If a task overrun , it effects only the same processor.
- No migration cost.
- A priority queue is maintained for each processor, so the queue length is small and queue access faster.
- The main advantages of using this approach for multiprocessor scheduling is that it reduces the multiprocessor scheduling to single processor scheduling.

The following disadvantages are observed

- The major disadvantage of the scheduling is allocation of the tasks to the processor. Finding the optimal assignment of tasks to processors is a bin-packing problem which is NP-Hard .
- Processor might remain idle while another processor might have scheduling overhead .

The performance of the partitioned scheduling algorithms depends on the bin packing algorithms used to partition the tasks among the processors of the multiprocessor system. In reality bin packing algorithm cannot provide task partitioning with total utilization more than $(m + 1)/2$ where m is the number of processor[7]. As a result, in the worst-case situation, the partitioned scheduling methods achieve slightly more than 50% of the multiprocessor system's processing capability for task execution, while the remaining nearly 50% remains idle.[5]

Methods for Task Partitioning:
The simplest bin packing method is Next Fit(NF). It just takes an item and places it in a bin until it is full, then moves on to the next bin and fills it again . This process is repeated until every object is allocated to a single bin. NF's complexity is $O(n)$

The first fit algorithm is a better but slower algorithm (FF). It is similar to NF, except before taking a new bin, it checks to see if there is another bin that suits the item. As a result, its complexity is $O(n * log(n))$. The best fit algorithm (BF) selects the bin with the least remaining capacity that nevertheless fits the item. It also has an $O(n * log(n))$ complexity.

You can refine these approaches by sorting the items by decreasing or increasing characteristics. This is frequently used for task sets with restricted and arbitrary deadlines. . There are a few more basic solutions for solving the bin packing issue, but these are the most essentials. The small tasks algorithm (ST) assigns jobs to processors with comparable harmonic periods. This method is very useful for activities with low utilization.

The generic task(GT) algorithm is a combination of two allocation algorithms. It divides tasks into two categories. One group had tasks with a utilization of more than 1/3 while the other had tasks with a utilization of less than or equal to 1/3. The ST method is used to allocate tasks to the first group, while the FF algorithm is used to give tasks to the second group. This approach has an $O(n)$ complexity and is utilized for a broader set of problems .

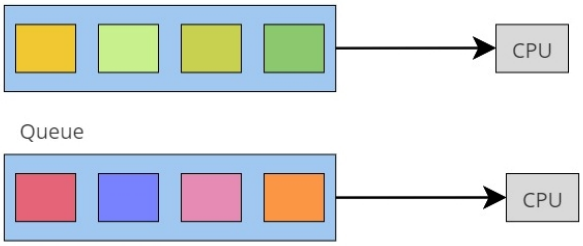

Fig. 5. Partitioned EDF scheduling [9]

B. Global EDF

In global scheduling all the ready tasks are placed in the global priority queue ,from which the scheduler executes the higher priority task. A task can be executed to any of the available processors even after preemption. Global EDF chooses the m highest priority ready tasks to run on the multiprocessor system's m processors at time t. [4] Some Advantages of global scheduling -

- Usually there are fewer context switches because a preemption will only occur when no other processor is idle.
- Effects of greater or less execution time then expected can be distributed to all other processors
- No allocation required
- The processor is always occupied with ready tasks.

The following disadvantages are observed

- A job might be often relocated from one processor to another,causing system migration overhead.
- For ready tasks, global scheduling uses a single queue. The queue length can be long and accessing the queue might require time.

For periodic task-sets Andersson et al. build an utilization bounds with implicit deadlines [7].it shows highest utilization constraint for any global fixed work priority algorithm is $(m+1)/2$. Thus, G-EDF and its variations have the same usage constraint as partitioned EDF[5].

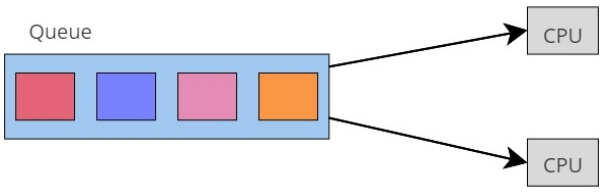

Fig. 6. Global EDF scheduling [9]

C. Hybrid Scheduling

Depending on the architecture, a task might migrate from one processor to another on global scheduling. Frequent migration causes high overhead , excessive communication loads and cache misses which increase the worst case execution time. These problems will not occur in the fully partitioned or non-migration scheduling approaches. On the other hand, in partitioned Scheduling the available processing capability is fragmented and a large fraction of processing capacity remains unused. The maximum utilization bound in a fully partitioned approach is just about 50% of the total processing capacity, so utilization is very low [4].

So overall due to all these disadvantages, cluster scheduling is developed which solves some of the major problems from global and partitioned scheduling .In this approach tasks are statically assigned to clusters, and then tasks are scheduled globally inside each cluster. There are two forms of cluster based scheduling :

- Physical cluster - the scheduling assigns processors statically to each cluster, with each cluster containing a subset of processors from the available multi-processor system.
- Virtual cluster - The processors in virtual cluster-based scheduling are dynamically mapped to each cluster, and thus the cluster and multiprocessors have a one-to-many relationship.[6].

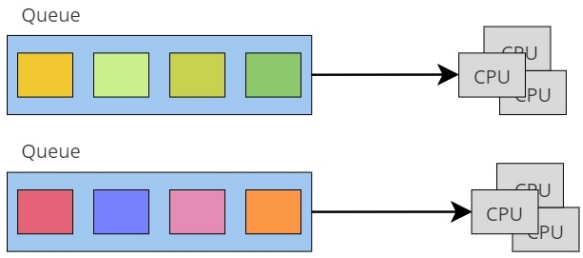

Fig. 7. Cluster scheduling [9]

Thus, clustering is a method of partitioning processors into a smaller number of subsystems to which tasks are assigned. As a result, capacity fragmentation is reduced in partitioned systems, and the number of processors in each cluster is lower than on the overall system, reducing global queue length and migration overhead. Despite migrations, processors in the same cluster may share the same cache, which minimizes the penalty of increasing worst-case execution time.

V. Conclusion

Earliest deadline first is a scheduling algorithm used for both unicore and multicore processors. In EDF scheduling we check the nearest deadline before scheduling a task. EDF can be used for both periodic and aperiodic tasks. EDF can be implemented on both preemptive and nonpreemptive scheduling . In a multi-core system tasks are very complex to schedule and it is usually NP hard. The main problems for scheduling on multicore are creating priority and allocations for active tasks. We talked about 3 types of scheduling approaches: they are Partitioned , Global and Clustered scheduling. In partitioned scheduling, no task migration is allowed. Tasks are scheduled to specific processors which work more like a collection of independent uniprocessor systems. In Global scheduling tasks can freely migrate and execute on any processor. In a large multi-core system these scheduling approaches have drawbacks and limited achievable processor utilizations. Partitioned scheduling needs to solve a bin-packing-like problem to assign tasks to processors because of these problems, restrictive caps on total utilization are generally required to ensure timing constraints, for both hard real-time (HRT) and soft real-time (SRT) systems. Global scheduling can have high overheads in reality due to congestion for the global run queue and non-negligible migration overheads among processors.Global scheduling is not an ideal choice for multi-core system Clustered scheduling is developed on the base of both partitioned and global scheduling. Clustered approach solves some of the major problems from P-EDF and G-EDF, here tasks are assigned to clusters and then inside a cluster tasks are scheduled globally. So Global scheduling is not a good approach for a real time system on multi-core. Partition scheduling has allocation and bin packing problems which are NP hard. In Clustered approach, many unanswered questions exist like, Will the chosen cluster size perform equally well for hard and soft real time systems? How does the impact of various preemption and migration related overheads compare to scheduling overheads[10]? Allocation , priority , bin problem and other disadvantages of each approach make multi-core scheduling a complex problem to solve compared to EDF in a uni-core system.

References

[1] Buttazzo, G. C. (2011). Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications (Real-Time Systems Series Book 24) (3rd ed.). Springer.

[2] Baruah, S., Bertogna, M., & Buttazzo, G. (2016). Multiprocessor Scheduling for Real-Time Systems (Embedded Systems) (Softcover reprint of the original 1st ed. 2015 ed.). Springer.

[3] John Carpenter, Shelby Funk, Philip Holman, Anand Srinivasan, James Anderson, and Sanjoy Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. Handbook on Scheduling Algorithms, Methods, and Models, pages, pages 30–1, 2004.

[4] Robert I. Davis and A. Burns. A survey of hard real-time scheduling for mul- tiprocessor systems. ACM Computing Surveys, 43(4):1–44, October 2011.

[5] Geoffrey Nelissen, Dragomir Milojevic, and Joël Goossens. Efficient Optimal Multiprocessor Scheduling Algorithms for Real-Time Systems. PhD thesis, PhD thesis, Université Libre de Bruxelles, 2013.
[6] Farhang Nemati. Partitioned Scheduling of Real-Time Tasks on Multi-core Platforms. School of Innovation, Design and Engineering, Mälardalen University,2010.
[7] Björn Andersson, Sanjoy Baruah, and Jan Jonsson. Static-priority scheduling on multiprocessors. In Real-Time Systems Symposium, 2001.(RTSS 2001).Proceedings. 22nd IEEE, pages 193–202. IEEE, 2001., 1961.
[8] Ahmed, Arwa. "Scheduling Algorithms in Operating Systems: Earliest Deadline First Scheduling." HubPages, discover.hubpages.com/technology/Scheduling-Algorithms-in-Operating-Systems-Earliest-Deadline-First-Scheduling.
[9] Völkel, F. Event Based Scheduling of Real-Time Multicore Systems.
[10] Bastoni, A., Brandenburg, B. B., Anderson, J. H. (2010, November). An Empirical Comparison of Global, Partitioned, and Clustered Multiprocessor EDF Schedulers. In RTSS (Vol. 10, No. 0, pp. 14-24).