Schedulability analysis of globally scheduled multiprocessor schedulers for real time system: A Review

Rahul Goradia#1, Dr. Apurva Shah#2

#1PhD Scholar, Computer Science and Engineering Department, M. S. University, India
#2Associate Professor & Director of Computer Center, M. S. University, India
rahul.goradia-cse@msubaroda.ac.in
apurva.shah-cse@msubaroda.ac.in

Abstract—In recent years, massive migration from single processor devices to multi-core computing devices has taken place in industry. General purpose and real time embedded systems have found a great market in multiprocessor systems. Researchers have suggested number of solutions to address computational demand on multiprocessor applications. So many scheduling policies have been designed and established to meet the real time requirements on multiprocessor systems. This paper provides the analysis of schedulability for periodic and sporadic tasks of Real-Time systems which is having Symmetric Multi-Processor (SMP). We will focus on job level dynamic priority EDF (global scheduling algorithm), where tasks can transfer from processor to processor during execution.

Keywords: Global Scheduling, Multiprocessor, Real Time Systems, Scheduling.

I. Introduction

Requirement of real time systems in multifaceted controller system has been increased very exponentially in recent years. As attempts of increasing the operating frequency of current uniprocessor computing devices cause serious problem of heating and huge amount of power consumption industry has moved towards implementing real time systems on SMP platform. Real-time multiprocessor systems are nowadays commonly used in embedded systems. Enough research has already been done for uniprocessor real time systems schedulers. Researchers have developed optimum uniprocessor algorithms of scheduling for sporadic and periodic real time tasks. RM, DM and EDF are found to be ideal algorithms for uniprocessor system. Applying the same algorithms on multiprocessor environment however do not gives same result. Investigating and refining uniprocessor algorithms to generate optimum results over multiprocessor platform has always been area of interest for researchers. Researchers have focused on two issues while dealing with multiprocessor scheduling: Processor selection and Priority assignment. Mainly global and strict partitioning are two main multiprocessor scheduling policies to select processor on which job will execute. In global scheduling approach job can run on any of the processor available. While in strict partitioning approach each and every task is assigned single processor to be executed and task is not permitted to run on other processor. Job migration is strictly not permissible in partitioning approach. Clustering and semi-partitioning approaches are hybrid approaches to select processor for task scheduling. In global approach task is allowed to execute on either a single processor or cluster of processors. In semi partitioned approach job level relocation is allowed on predefined set of processors. While global, partitioning, clustering and semi-partitioning approaches contribute in selecting the available free processor, setting priority of a task/job is altogether a different problem. Unlike uniprocessor platform RM, DM and EDF are not found to be as optimum priority assignment method on multiprocessor platform. Many heuristics of these algorithms are used and they exhibit different pattern from each other. Pfair policy of priority assignment is claimed to be optimum in case of global scheduling approach. Many factors have been identified by researchers to measure performance of algorithm. Amongst them utilization ratio, utilization bound and schedulability are mainly used parameters to access performance of a scheduler. In this paper we present analysis for schedulability of globally scheduled multiprocessor schedulers. In first section we describe task-model which is commonly used in real time system and define parameters used to evaluate performance. Next section describes algorithms for global scheduling. In second section we represent global scheduling algorithms. In this section we present schedulability analysis tests for global scheduling algorithm global-EDF.
II. Task Model

Liu and Layland developed simplest prototype for real time sporadic process [1]. In this model, a sporadic task \( \tau_i \) is categorized by three factors: a worst-case execution requirement equal to \( C_i \) time unit, a comparative deadline \( D_i \) period unit later its arrival time, and a successive job-arrivals separation period \( T_i \) time unit which is also referred as period of task. Liu and Layland task-model can be signified by \( \tau_i = (C_i, D_i, T_i) \). Such a sporadic task represents an endless arrangement of jobs. We refer to the interval, of size \( D_i \), between such a job’s arrival instant and deadline as its window of scheduling. A task of sporadic system is comprised of several such sporadic tasks. A pool of sporadic tasks is mentioned as a system of sporadic task and is denoted as \( \tau_i. \) Task system can be signified by \( \tau = \{\tau_1, \tau_2, \ldots, \tau_n\} \), with \( \tau_i = (C_i, D_i, T_i) \) for all \( i, 1 \leq i \leq n \). Task system \( \tau \) is said to be a controlled task system if it is assured that each and every task \( \tau_i \in \tau \) has its comparative deadline factor no greater than its period: \( D_i \leq T_i \), and an implicit-deadline task system if \( D_i = T_i \) for all \( i \in \tau \). (Implicit-deadline systems are also known as Liu and Layland [1] task systems.) In this paper, our attention is limited to controlled and implicit-deadline task systems. We also assume aentirely preemptive execution model where any executing job may be interjected at any instant in time, and its execution can be resumed later with no cost or penalty. In this paper, focus is only on collection of independent tasks, all with hard real-time requirement, and preemptable scheduled with symmetric multiprocessors using global algorithm.

Task and System Characteristic

Utilization is the most important concept used in analysis of systems of sporadic task on multiprocessor. The utilization \( u \) of a task \( \tau \) is the ratio \( C_i / T_i \) of its completing time \( C_i \) to its duration \( T_i \). The total utilization \( u_{\text{total}}(\tau) \) is summation of utilization of all tasks and the largest utilization \( u_{\text{max}}(\tau) \) of a system of tasks \( \tau_i \) is defined as maximum \( u \) of all the task from task system \( \tau \). In similar manner density utilization \( \delta \) of a task \( \tau \) is the ratio \( C_i / D_i \) of its completing time \( C_i \) to its time limit \( D_i \). The total density \( \delta_{\text{total}}(\tau) \) is summation of density of all tasks and the largest density \( \delta_{\text{max}}(\tau) \) of a system of tasks \( \tau_i \) is defined as maximum \( \delta \) of all the task from task system \( \tau \). Another important parameter used for schedulability analysis is Demand Bound Function (DBF). For any slot length \( t \), demand bound function DBF(\( \tau \), \( t \)) of task \( \tau \) confines the extreme aggregate execution necessity by jobs. Demand bound function of a job \( \tau \) that both arrive in, and deadlines within any interval of length \( t \) can be defined as [2]

\[
DBF(\tau, t) = \max \left( 0, \left( \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i \right)
\]

Schedulability

System of real-time task is said to be \( A \)-schedulable in compare to a given scheduling algorithm \( A \), if the algorithm \( A \) schedules the system such that any job of any task will not miss deadline and all jobs of all tasks will successfully execute, under all acceptable mixtures of job-arrival arrangements by the dissimilar tasks covering the system. A schedulability test for scheduling algorithm \( A \) takes as input the configuration of a real-time system, and determines whether the system is \( A \)-schedulable or not. An \( A \)-schedulability test is said to be exact if it correctly identifies all \( A \)-schedulable systems, and sufficient if it may fail to identify some \( A \)-schedulable systems (it must guarantee, though, that all identified systems are indeed \( A \)-schedulable). A task-system \( \tau \) is schedulable by an algorithm \( A \) if \( A \) confirms that the timing restrictions of all available tasks in \( \tau \) are metpositively. \( \tau \) is said to be possible under a class \( C \) of scheduling algorithms if \( \tau \) is schedulable by some algorithm \( \mathcal{A} \in \mathcal{C} \). An algorithm \( A \) is said to be ideal regarding class \( C \) if \( \mathcal{A} \in \mathcal{C} \) and Appropriately schedules each and every task system that is feasible under \( C \). When the class \( C \) is not specified, it should be assumed to include all possible scheduling algorithms.

III. Global Scheduling Algorithms

A global scheduling algorithm maintains a system wide queue in which ready tasks are arranged in a priority order. Priority of task is assigned by policies based on priority like RM, EDF etc. Whenever processor becomes idle the job having priority highest is scheduled on available processor. Every time a task with higher priority than tasks executing on the processors is released lowest priority task amongst accomplishing tasks is preempted back to ready queue and higher priority task is scheduled. Later if a processor becomes idle, preempted job may be scheduled on that available processor though newly available processor is different than the processor from which it was preempted. Thus a task can migrate over available processors. This mechanism guarantees of executing m-highest priority task for multiprocessor platform having m processors.

While in view of global scheduling, schedulers may be approximately distributed in three groups according to the priority of a task which has for the duration of its execution. If the priority of a task cannot modify during the entire task lifespan, the scheduling algorithm having "fixed task priority". If the priority can modify only at job margins, as with EDF, then the algorithm will be having "fixed job priority". The above classes are frequently stated as "priority driven". Lastly, if the priority of task can modify also in the duration of the job accomplishment, as for the P-fair class of algorithms mentioned in [3], then the algorithm has "(fully) dynamic priority". Algorithms from the final class can have a greater consumption bound, reaching the number of
processors when deadlines are identical to periods. On the second side, they have a greater number of preemptions and relocations and a more difficult implementation. Due to these causes, it may be more promising to use a priority driven scheduler that has all the mentioned advantages related to the global scheduling.

Study of global schedulers on multiprocessor environment is very much complex compared to that of uniprocessor platform. Hong and Leung proved that without knowing future arrival, period and deadline it is not possible to build an online scheduler for more than one processor[4]. Dertouzos and Mok proved that without previous knowledge of start-time, computation time and deadline no scheduling algorithm can be ideal for environment collected by two or more than two processors [5].

When considering dynamic-job priority scheduling algorithms, Pfair algorithms [6], [7] are optimal for implicit time limit periodic and sporadic task systems, allowing a schedulable utilization equal to the available system capacity.

IV. Global Algorithm Schedulability

For guaranteed schedulability in m multiprocessor system load of all task should be less than m is the necessary condition but not sufficient. The global multiprocessor scheduling of implicit-deadline (Liu and Layland) sporadic task systems was studied in [8]. For m multiprocessor system adequatesituation for implicit-deadline sporadic task system r to be global EDF schedulable is u_sum(r) ≤ m - (m - 1) u_max(r) . To this schedulability condition minor extension can be applied and density test [8] for global EDF schedulability of controlled sporadic task systems achieved is δ_sum(r) ≤ m - (m - 1) δ_max(r). Baker[9], [10], Bertogna et al.[11] and Baruah et al. [12] each designed test for global EDF schedulability analysis.

The Goossens, Funk and Baruah Test (GFB Test)

Let SMP platform π consist of mequivalent processors and each has computing capacity s_p. If a job executes on the i-th processor for t time units than it completes s_p*t units of its total execution time. Let S_p and s_p be the sum of the computing capacities of all processors and the computing capacity of the fastest processor of platform π, respectively. The relation between an optimal algorithm for a SMP platform and global EDF scheduling[13] can be shown as S_p = Σ_t∈π U_t and s_p = U_max. If m ≥ s_p = 1 - s_p than set of jobs can be scheduled with global EDF on SMP having unit capacity multiprocessors. Relating all these results A periodic task set π is EDF-schedulable upon SMP system composed by m processors with unitary capacity, if

\[ u_{\text{sum}}(r) \leq m(1 - u_{\text{max}}) + u_{\text{max}} \]

This result is obtained for implicit deadline task having period same as deadline. The same result was extended for task having deadlines less than or equal to periods which states the condition of schedulability of sporadic task set π with global EDF upon SMP system composed by m processors with unitary capacity, that is

\[ δ_{\text{sum}}(r) \leq m - (m - 1) δ_{\text{max}}(r) \]

BAKER test

Baker used another approach to derive sufficient schedulability condition[10]. Assuming that task r_k misses the deadline d_i. If job arrived at t_i time than between [r_k, d_k] time duration load should be at least m - (m - 1) δ_k(r) and all processors should be executing job other than r_k for D_k - C_k time. If during [r_k, d_k] load is less than δ_sum(r) schedulability is guaranteed. Analyzing loads in[r_k, d_k] is not always possible, so Baker increased this observation interval such that calculation of carry in of interfering task is possible. Here carry in time is the time consumed by job instance before r_k between [r_k, d_k]. So Baker used largest possible interval [a, d_k], called busy window, so that load is greater than mm - (m - 1) δ_k(r). So overall proposed condition for task set π having n task with global EDF on SMP system for guaranteed schedulability is

\[ \forall r_k: \sum_{i=1}^{n} \min(1, β_i) \leq m(1 - λ_k) + λ_k, \]

where

\[ β_i = \begin{cases} U_i \left(1 + \frac{T_i - D_i}{D_k}\right) & \text{if } λ_k \geq U_i \\ U_i \left(1 + \frac{T_i - D_i}{D_k}\right) + \frac{C_i - λ_k T_i}{D_k} & \text{if } λ_k < U_i \end{cases} \]

EDF-US Test

Srinivasan and Baruah[14] studied the global EDF scheduling for periodic tasks on multiprocessors. Using hybrid scheduling policy they suggested a new method to deal with tasks having high utilization. In their hybrid approach they defined a constant ξ and give fixed highest priority to the task having higher utilization than ξ. Tasks with lower utilization than ξ were scheduled with priority assigned by conventional EDF
algorithm. This algorithm is called EDF-US[ξ]. In this research they showed that any system of independent periodic tasks for which the consumption of each and every specific task is at most \( m/(2m-1) \) can be scheduled positively on \( m \) processors if the total utilization is at most \( m^2/(2m - 1) \).

**BCL Test**

Bertogna,Cirinei and Lipari used very similar ideas to that used in [9]and made extraperfections in global EDF schedulability tests. In BCL test analysis start time set to \( t_a - D_k \), which is different than BAK test. They observed that the proof of the utilization bound test of [15] extends naturally to cover pre-period deadlines if the utilization test \( u_i \) is replaced by \( c_i/D_i \). The same proof extends to the case of post-period deadlines if \( c_i/D_i \) is replaced by \( \lambda_i = c_i/\min \{D_i,T_i\} \). Bertogna et al came with schedulability test which suggests constraint deadline task set is global EDF schedulable on SMP if for each task \( \tau_k \) one of the following is true

\[
\sum_{i \neq k} \min(\beta_i, 1 - \lambda_k) < m(1 - \lambda_k) \\
\sum_{i \neq k} \min(\beta_i, 1 - \lambda_k) = m(1 - \lambda_k) \text{ and } \exists i \neq k : 0 < \beta_i < 1 - \lambda_k
\]

where

\[
\beta_i = N_i c_i + \min\{c_i, \max\{0, d_k - N_i T_i\}\}/d_k
\]

and

\[
N_i = \left[ d_k/d_i \right] - 1
\]

Bertogna observed that any task \( \tau_k \) failures to meet a deadline due to interference of other tasks, and the extreme segment of the workload of any job that can give to their inferences is \( 1 - \lambda_k \).

**V. Conclusion**

Global scheduling can have higher overhead at least two cases: the contention delay and the synchronization overhead. Overhead of synchronization for a singletransmitting queue is greater than for processor queues. The cost of continuing a task may be greater if it is on a distinct processor than on the processor where it last accomplished because in global scheduling we allow migration of tasks on other processors. The latter cost can be quite variable, since it depends on the actual portion of a task’s memory that remains in cache when the task resumes execution, and how much of that remnant will be referenced again before it is overwritten. Backer test implements more sophisticated analysis than BCL test. BCL, GBF and BAK tests are not comparable. BCL tests performs beers in case of high utilizations tasks. BCL can discover more schedulable tasks than GBF and BAK.

**VI. References**

[1] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment,” J. ACM, vol. 20, no. 1, pp. 46–61, Jan. 1973.
[2] S. K. Baruah, A. K. Mok, and L. E. Rosier, “Preemptively scheduling hard-real-time sporadic tasks on one processor,” in [1990] Proceedings 11th Real-Time Systems Symposium, 1990, no. December, pp. 182–190.
[3] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. a. Varvel, “Proportionate progress: A notion of fairness in resource allocation,” Algorithmica, vol. 15, no. 6, pp. 600–625, 1996.
[4] K. S. Hong and J. Y.-T. Leung, “On-line scheduling of real-time tasks,” IEEE Trans. Comput., vol. 41, no. 10, pp. 1326–1331, 1992.
[5] M. L. Dertouzos and A. K. Mok, “Multiprocessor online scheduling of hard-real-time tasks,” IEEE Trans. Softw. Eng., vol. 15, no. 12, pp. 1497–1506, 1989.
[6] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. a. Varvel, “Proportionate progress: A notion of fairness in resource allocation,” Algorithmica, vol. 15, no. 6, pp. 600–625, Jun. 1996.
[7] J. H. Anderson and A. Srinivasan, “Mixed Pfair/ERfair scheduling of asynchronous periodic tasks,” in Proceedings - Euromicro Conference on Real-Time Systems, 2001, vol. 68, no. 1, pp. 76–85.
[8] J. Goossens, S. Funk, and S. Baruah, “No Title,” Real-Time Syst., vol. 25, no. 2/3, pp. 187–205, 2003.
[9] T. P. Baker, “Multiprocessor EDF and deadline monotonic schedulability analysis,” in Proceedings. 2003 International Symposium on System-on-Chip (IEEE Cat. No.03EX748), 2003, no. Dm, pp. 120–129.
[10] T. P. Baker, “Multiprocessor EDF and deadline monotonic schedulability analysis,” RTSS 2003. 24th IEEE Real-Time Syst. Symp. 2003, no. Dm, 2003.
[11] M. Bertogna, M. Cirinei, and G. Lipari, “Improved Schedulability Analysis of EDF on Multiprocessor Platforms,” in 17th Euromicro Conference on Real-Time Systems (ECRTS'05), 2010, vol. 46, no. 1, pp. 209–218.
[12] S. Baruah and T. Baker, “Schedulability analysis of global edf,” Real-Time Syst., vol. 38, no. 3, pp. 223–235, Apr. 2008.
[13] S. Funk, J. Goossens, and S. Baruah, “On-line scheduling on uniform multiprocessors,” in Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420), pp. 183–192.
[14] A. Srinivasan and S. Baruah, “Deadline-based scheduling of periodic task systems on multiprocessors,” Inf. Process. Lett., vol. 84, no. 2, pp. 93–98, Oct. 2002.
[15] J. Goossens and R. Devillers, “Feasibility intervals for the deadline driven scheduler with arbitrary deadlines,” in Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA ’99 (Cat. No.PR00306), 1999, pp. 54–61.