Guaranteeing Schedulability of Splittable Hard Real-Time Tasks for Non-Preemptable Devices

Mitra Nasri
Chair of Real-time Systems,
Technische Universität Kaiserslautern,
Germany
nasri@eit.uni-kl.de

Gerhard Fohler
Chair of Real-time Systems,
Technische Universität Kaiserslautern,
Germany
fohler@eit.uni-kl.de

Nafiseh Moti
School of Electrical & Computer Engineering,
University of Tehran, Tehran,
Iran
n.moti@ut.ac.ir

***Abstract*—Nowadays it has been possible for many embedded systems to use peripheral computational resources such as graphics processing units (GPUs) to execute hard real-time data-parallel applications. However, because of the contentions on the buses and memory access channels, or the limited interface of the devices such as GPUs, the executing application must not be preempted while it is occupying the resource. Since non-preemptive scheduling of hard real-time periodic tasks is an NP-Hard problem, one efficient solution is to split the task into non-preemptive chunks. In this paper we introduce fine-grained periodic resource model (FG-PRM) which is customized for non-preemptable devices. Using the notion of reservation task model, we derive schedulablity condition as a function of the resource period, then we solve it by applying mathematical function estimation. The resulting parameter assignment method have been evaluated by its acceptance ratio. We have demonstrated the efficiency of our method through the experiments.**

I. Introduction

Nowadays the use of multi-core and many-core architectures as powerful computational resources has been increased in embedded and cyber-physical systems [1]. Using these resources it is possible to have hard real-time applications with significant computational requirements such as emergency collision avoidance in automatic cars and intelligent cruise control [2] running on the many-core peripheral computational devices such as graphics processing units (GPU). Many of these applications are data-parallel and they can be split into smaller chunks [3].

For many reasons such as contentions on the buses and memory access channels, or the use of GPUs which communicate with the processor through interrupt based interface, the executing application must not be preempted while it is occupying the resource [1], [4]. However, guaranteeing schedulability of a non-preemptive system with periodic tasks is an NP-Hard problem [5]. One efficient solution is to split the task into non-preemptive chunks. Many applications such as those consisting of matrix multiplications [3], [2] support flexible split sizes, however, it has to be performed at design time. One of the pioneer solutions to split such tasks for guaranteeing soft deadlines for non-preemptive accessible devices like GPUs has been introduced in [6], however, to the best of our knowledge, there is no solution to guarantee hard deadlines.

In this paper we use the notion of periodic resource model (PRM) [7] and make it customized for non-preemptable devices. PRM describes the resource behavior as a periodic model with two phase; the resource is available for duration C then it becomes idle. This pattern will be repeated periodically with period P. To determine high priority tasks to be executed during the execution budget within the active phase, PRM uses an ordinary scheduler such as RM which might result in task's preemptions because the length of active phase might be greater than period of some of the tasks. Moreover, the cost of exact schedulability analysis of this model is relatively high [8] and current approximated solutions such as [9] are not efficient for all task sets, specially when the ratio between the shortest and the longest period is large. In those cases, outputs of [9] and [8] will produce pairs $C \approx P$ while the actual required resource might be far less than the resulting parameters. Due to these drawbacks, the existing feasible parameter assignment methods for PRM are not computationally affordable or efficient for our target systems.

PRM can be considered as a special type of server based solutions which has one periodic task to handle all other tasks in the system. Although the traditional usage of these approaches was to effectively execute the aperiodic tasks [10], by the introduction of constant bandwidth servers [11], the temporal partitioning model [12], and the reservation task model [13], these methods have been extended to provide isolation for the tasks or to handle sporadic tasks as periodic ones [13]. However, because of the parameter assignment methods applied in the most of these studies, tasks might be preempted inside the execution budget of the server task or because of the preemptions caused by other periodic tasks.

In this paper, we introduce fine-grained periodic resource model (FG-PRM) as a customized PRM that guarantees non-preemptive execution inside the active phase. To construct the basic parameter assignment problem as a function of the PRM parameters, i.e., C and P, we use the formulations of the reservation task model [13] which assigns one server to one task. However, to construct our customized PRM, we represent the task set with only one reservation task. Thus, in every period, each task has one portion of execution with a fixed length, and the tasks finish their execution after a fixed number of releases of the reservation task. To solve the parameter assignment problem we use mathematical function approximation. In summary, in our method, the schedulability of each task is guaranteed because of the use of the reservation task model formulation, while non-preemptive execution is guaranteed by assigning fixed non-interleaving execution windows to the tasks.

Our solution, not only helps GPU applications but also provides a mean to feasibly schedule any other application on non-preemptable I/O devices as long as flexible splitting is permitted. It can be applied on periodic or sporadic tasks with arbitrary release offsets or explicit deadlines shorter than period. Besides, our approach does not need real-time scheduling algorithms; it provides a prioritized queue of the tasks with specified execution budget for each task within which the task can be executed non-preemptively. Size of these budgets (or splits) is obtained as a parameter of the model at design time, hence, it is available for the applications beforehand. Moreover, we are able to obtain the upper bound of the response time of the tasks and its error bound in $O(1)$.

The reminder of the paper is organized as follows; Sect. II introduces the system model. In Sect. III, the concept our work is presented which is followed by parameter assignment method in Sect. IV. Few experimental results have been presented in Sect. V and the paper is concluded in Sect. VI.

II. System Model

We consider a set of hard real-time independent sporadic tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task is identified by $\tau_i : (r_i, c_i, T_i, d_i)$, where r_i is the release offset, c_i is the execution time, T_i is the minimum inter-arrival time (period), and $d_i \leq T_i$ is the relative deadline of the task $\tau_i \in \tau$. The tasks have been indexed according to their period such that $T_i \leq T_{i+1}$, $1 \leq i < n$. Since we assume non-preemptable devices, c_i can be considered as resource occupation time, however, it has to be possible that we split c_i into smaller values at least at design time. The device is able to be used by each application for a specified amount of time, however, during this time, it cannot be preempted. GPUs are an example for such devices [2]. When a task enters GPUs, it will not leave it until end of its execution. Then GPU device notifies the operating system using an interrupt based interface. Although it is possible to split the tasks into smaller chunks and run those chunks on GPU device non-preemptively, size of the splits has to be known at compile time as mentioned in [6].

III. Fine-Grained Periodic Resource Model

Fine-grained periodic resource model governs the device as a PRM with period P and budget C. In the active phase, a queue of tasks is gradually sent to the device each of them has o_i units of device occupation time (DOT). The order of the tasks in the queue can be either fixed or dynamic, yet none of these options has any impact on the schedulability. For the simplicity we assume tasks will occupy the device according to their index from τ_1 to τ_n unless they are not released before their dedicated dispatch time. The later case happens when the original task is not in the system because of the assumptions regarding explicit deadlines and sporadic releases. If such tasks come later than their assigned window, they can be dispatched to the device otherwise their window will be simply ignored.

FG-PRM is based on reservation tasks [13] where for each original task, one reservation task is assigned. Theis and Fohler [13] have proven that if the parameters of the reservation task are selected in the following way, it can always guarantee schedulability of the original task as long as the reservation tasks can be feasibly scheduled by some the underlying scheduling algorithm.

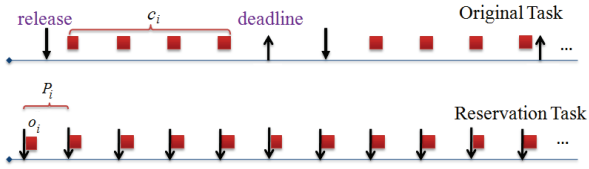

Fig. 1. An example of task execution using reservation task model [Theis11]

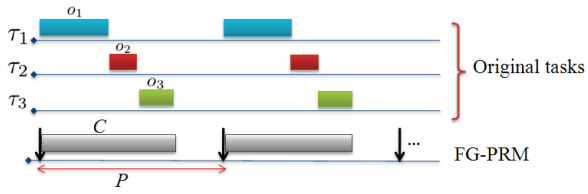

Fig. 2. An example of execution using FG-PRM

$$o_i = c_i / k_i \tag{1}$$

$$p_i = \frac{d_i + o_i}{k_i + 1} \tag{2}$$

where $k_i \in \mathbb{N}^+$ is the number of releases of the reservation task under one release of the original task, and p_i is the period of the reservation task. Fig. 1 shows the worst case scenario happened in the execution of the original task because the first release of the reservation task cannot be used to execute the original task. Besides, in our model (shown by Fig. 2), tasks are not allowed to be split at run time, hence, they can be executed only if they are released before the scheduled window of the reservation task. Consequently, we have to consider $k_i + 1$ releases with at least k_i effective releases.

If all tasks are scheduled by one reservation task, we can replace p_i by P in (2), then k_i and C are obtained as

$$k_i = \begin{cases} d_i/P & d_i/P \in \mathbb{N} \\ \lfloor d_i/P \rfloor - 1 & otherwise \end{cases} \tag{3}$$

$$C = \sum_{i=1}^{n} c_i / k_i \tag{4}$$

The resource utilization U^R is obtained by using the worst case value of (3) in (4) as

$$U^R = \frac{1}{P} \sum_{i=1}^{n} \frac{c_i}{k_i} = \frac{1}{P} \sum_{i=1}^{n} \frac{c_i}{\lfloor \frac{d_i}{P} \rfloor - 1} \tag{5}$$

Theorem 1. *Task set τ is schedulable by FG-PRM with period P and budget C if $U^R \leq 1$.*

Proof Having $U^R \leq 1$, all tasks have their dedicated time slot in every active phase. Also according to (1) each original task is guaranteed to have $k_i o_i = c_i$ DOT before its deadline, hence, it can be feasibly scheduled. ■

Fig. 3 shows U^R as a function of P for one task in two cases. According to this figure, U^R increases with the increase

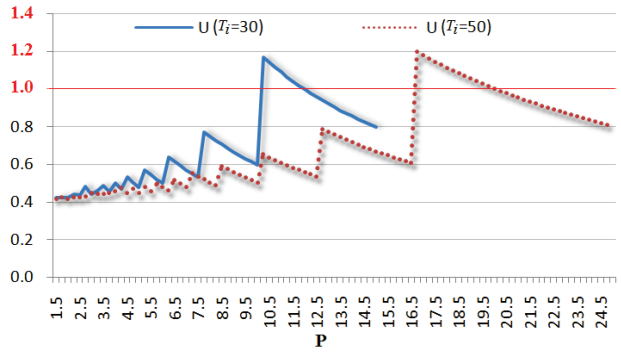

Fig. 3. U^R as a function of P for one task in two cases; $T_i = d_i = 30$ and $T_i = d_i = 50$ and $u_i = 40$

in P. Also it grows fast if tasks have small d_i. If P tends to 0, U^R tends to $U = \sum \frac{c_i}{d_i - 1}$ since

$$\lim_{P \to 0} \frac{1}{P} \sum_{i=1}^{n} \frac{c_i}{\lfloor \frac{d_i}{P} \rfloor - 1} = \lim_{P \to 0} \sum_{i=1}^{n} \frac{c_i}{P \lfloor \frac{d_i}{P} \rfloor - P} \leq \sum_{i=1}^{n} \frac{c_i}{d_i - 1} \tag{6}$$

Since many of our target applications have considerable computational requirements, they usually have large periods as well. Yet it has to be mentioned that maximum value of P in our model is $0.5d_1$ and it happens when $k_1 = 2$. As shown by Fig. 1, response time of the tasks will be $WCRT_i = d_i$ and its error is

$$WCRT_i^{err} \leq 2p_i - o_i \tag{7}$$

because in the best case scenario, the execution of the original task starts in the first release of the reservation task and finishes after o_i DOT after the release of the second-to-last reservation task. Both WCRT and its error are obtained in $O(1)$. Since WCRT of the task is equal to their deadlines, our model is efficient for the hard real-time applications which are not sensitive to long response times as long as deadlines are guaranteed. Moreover, in our work the overheads of the task splitting has been ignored. As a future work, we will apply [13] formulation to consider extra overheads of splitting into the utilization formulation. It is worth mentioning that our solution might not be efficient for dense task sets with tight deadlines and wide period values since it might have many ineffective reservations.

IV. Parameter Assignment Method

In this section we solve the parameter assignment problem. Upper and lower bounds of (5) can be obtained when $\lfloor \frac{d_i}{P} \rfloor$ is replaced by $\frac{d_i}{P} - 1$ and $\frac{d_i}{P}$ respectively. Thus we have

$$U^{up} = \frac{1}{P} \sum_{i=1}^{n} \frac{c_i}{\frac{d_i}{P} - 2} = \sum_{i=1}^{n} \frac{c_i}{d_i - 2P} \tag{8}$$

Fig. 4 illustrates the upper bound (8) for one task. Using Theorem 1 we obtain the schedulability problem as

$$U^{up} \leq 1 \Rightarrow \sum_{i=1}^{n} \frac{c_i}{d_i - 2P} \leq 1 \tag{9}$$

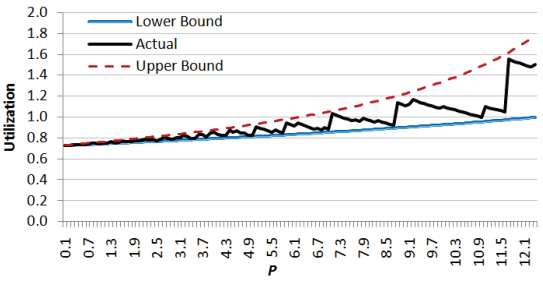

Fig. 4. U^{up}, U^R, and U^{low} as a function of P for a task set with 3 tasks $\tau_1 : (0, 12, 35, 35)$, $\tau_2 : (0, 10, 55, 55)$, and $\tau_3 : (0, 20, 99, 99)$.

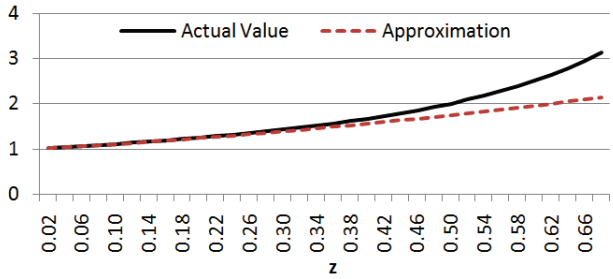

Fig. 5. The actual value of $\frac{1}{1-z}$ and its approximated value using the first 3 terms of (10).

However, it is not easy to solve (9) because the variable P appeared in the denominator of a sum. To make it solvable we use the following equality in Geometric series

$$\frac{1}{1-z} = 1 + z + z^2 + z^3 + ... \tag{10}$$

where $z < 1$. To keep our formulation solvable, we omitting degrees higher than 3 and only use from $1 + z + z^2$ terms of the series. Fig. 5 shows to what extend we can rely on the first 3 terms of the series. From this figure it is possible to deduce that if $\frac{2P}{d_1} \leq 0.6$, the error remained below 1. It leads to $P < 0.3d_1$. We rewrite (9) as

$$\sum_{i=1}^{n} u_i \frac{1}{1 - \frac{2P}{d_i}} \leq 1 \Rightarrow \sum_{i=1}^{n} u_i (1 + \frac{2P}{d_i} + \frac{4P^2}{d_i^2}) \leq 1 \tag{11}$$

where $u_i = c_i/d_i$. Thus we have

$$4P^2 \sum_{i=1}^{n} \frac{u_i}{d_i^2} + 2P \sum_{i=1}^{n} \frac{u_i}{d_i} + \sum_{i=1}^{n} u_i - 1 \leq 0 \tag{12}$$

Since u_i and d_i are known, it is easy to calculate values of the sums so we are able to simplify (12) as $aP^2 + bP + c \leq 0$. Since c is $\sum_{i=1}^{n} u_i - 1$ and it is always negative, $\Delta = b^2 - 4ac$ is positive, hence, resulting values for P are real numbers, and the inequality is solvable. However, one of the roots is always negative and is not acceptable because $0 < P < 0.3d_1$. Hence, the only valid value for P is

$$P \leq \frac{-2 \sum_{i=1}^{n} \frac{u_i}{d_i} + \sqrt{\Delta}}{8 \sum_{i=1}^{n} \frac{u_i}{d_i^2}} \tag{13}$$

To obtain feasible solution, first we calculate P according to (13). If P is larger than $0.5d_1$ and U^R, which is calculated

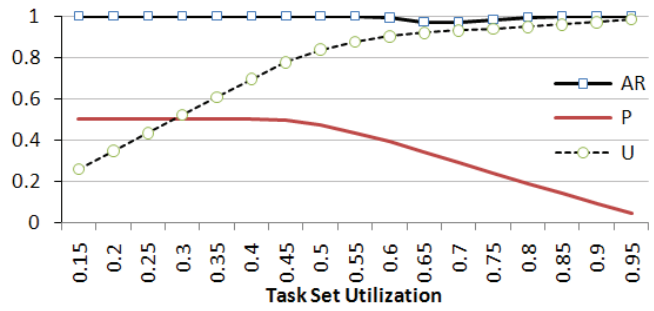

Fig. 6. Admission ratio, normalized period, and resource utilization as a function of task set utilization

from (5), is greater than 1, we try $P = 0.5d_1$. This situation happens in cases where the task set utilization is low and (13) produces relatively large values of P which are not compatible with our other constraints. In this case, we set $P = 0.5d_1$ and check whether U^R is smaller than 1 or not.

If a feasible P is obtained, it is possible to find k_i from (3), o_i from (1), C from (4). Computational complexity of this calculation is $O(n)$ since we have to calculate values of a, b, and c from (12). It is worth mentioning that although P might be close to $0.3d_1$, for other tasks with $d_i \gg d_1$, the ratio between P and d_i is small, hence, $(P/d_i)^3$, $(P/d_i)^4$, etc. become very small values which can be ignored during the calculation of (9). According to Fig. 5, one of the draw backs of our method is that our estimation of $\frac{1}{1-z}$ is not the upper bound. In the future work we try to find an upper bound for this function.

V. Experimental Results

In this section we evaluate the efficiency of FG-PRM regarding admission ratio of the task sets (AR), resource utilization C/P, and normalized resource period, i.e., $\frac{P}{d_1}$. Parameter of the experiments is the utilization of the task set which ranges from 0.15 to 0.95 with step 0.05. For each utilization, we generate 10000 random task sets each with 10 tasks, where their utilization is obtained from uUniFast algorithm. To bound the hyperperiod of the tasks we have considered fixed value $H = 10000$ and we have uniformly chosen value $f_i \in \{1, 2, \ldots, 100\}$ to be able to compute $T_i = H/f_i$, $c_i = u_i T_i$, and $d_i = T_i$.

The average results of AR, P, and U for all utilization values and for all generated task sets have been reported in Fig. 6. AR is obtain by dividing the number of feasible task sets by total number of generated task sets. Horizontal axis of this diagram is task set utilization. As shown by this figure, our method has a significant admission ratio with average 0.99. Also it highly utilizes the resource. In this figure, P is constant before utilization 0.45, and has the maximum value of $P = 0.5d_1$ which leads to normalized P equal to 0.5. From utilization 0.45, P start to decrease which means that condition (13) is activated. The fall of P continues in high utilization task sets. On the other hand, before utilization 0.45 where the period of the resource is constant, by the increase in the utilization of the task set, resource utilization increases linearly.

VI. Conclusion

In this work, a feasible solution for scheduling hard real-time splittable tasks have been presented which is well suited for interactions with non-preemptable devices. The solution is based on the notion of periodic resource model and we have focused on feasible parameter assignment for this model. Using the formulation of reservation task model we have reformulated the schedulability problem. Then the formula is approximated by a solvable polynomial of degree 2. The experiments shown the efficiently of the solution. As a future work, we consider overheads of splitting into account. Also we perform actual experiments on the GPU devices to show the efficiency of our solution on real benchmark applications.

Acknowledgment

The authors would like to thank Mehdi Ghasemi for his helpful comments. This work has been supported by the DAAD scholarship.

References

[1] G. A. Elliott and J. H. Anderson, "Globally scheduled real-time multiprocessor systems with gpus," *Real-Time Systems*, vol. 48, no. 1, pp. 34–74, 2012.

[2] G. Elliott and J. Anderson, "Real-world constraints of gpus in real-time systems," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, vol. 2, 2011, pp. 48–54.

[3] D. Grewe and M. OBoyle, "A static task partitioning approach for heterogeneous systems using opencl," in *Compiler Construction*, ser. Lecture Notes in Computer Science, J. Knoop, Ed. Springer Berlin Heidelberg, 2011, vol. 6601, pp. 286–305.

[4] A. Bastoni, "Cache-related preemption and migration delays: Empirical approximation and impact on schedulability," in *International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, 2010, pp. 33–44.

[5] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of period and sporadic tasks," in *IEEE Real-Time Systems Symposium (RTSS)*, 1991, pp. 129–139.

[6] C. Basaran and K.-D. Kang, "Supporting preemptive task executions and memory copies in gpgpus," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2012, pp. 287–296.

[7] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *IEEE International Real-Time Systems Symposium (RTSS)*, 2003, pp. 2–12.

[8] N. Fisher and F. Dewan, "A bandwidth allocation scheme for compositional real-time systems with periodic resources," *Real-Time Systems*, vol. 48, no. 3, pp. 223–263, 2012.

[9] N. Fisher and F. Dewan, "Approximate bandwidth allocation for compositional real-time systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2009, pp. 87–96.

[10] T.-H. Lin and W. Tarng, "Scheduling periodic and aperiodic tasks in hard real-time computing systems," *SIGMETRICS Performance Evaluation Review*, vol. 19, no. 1, pp. 31–38, 1991.

[11] L. Abeni, G. Lipari, and G. Buttazzo, "Constant bandwidth vs proportional share resource allocation," in *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE Computer Society, 1999, pp. 107–111.

[12] L. Almeida and P. Pedreiras, "Scheduling within temporal partitions: Response-time analysis and server design," in *ACM International Conference on Embedded Software (EMSOFT)*. ACM, 2004, pp. 95–103.

[13] J. Theis and G. Fohler, "Transformation of sporadic tasks for off-line scheduling with utilization and response time trade-offs," in *International Conference on Real-Time and Network Systems (RTNS)*, 2011, pp. 119–128.