JUET Research Journal of Science & Technology
5 (2), July 2019, 19-22
http://www.publishingindia.com/juet

# Efficient Real-Time Scheduling Analytics for Preserving Sporadic Server Budget

Ajitesh Kumar[1*] and S. K. Gupta[2]

[1]Research Scholar, AKTU Lucknow and Assistant Professor, GLA University Mathura, Uttar Pradesh, India. Email: ajitesh.kumar@gla.ac.in
[2]Associate Professor, Computer Science Department, BIET Jhansi, Uttar Pradesh, India. Email: guptask_biet@rediffmail.com
*Corresponding Author

**Abstract: Nowadays, real time system plays an important role in daily life. In real time systems, besides logical correctness, the timeliness of results is equally important. A real time system aims at computing the desired result within a prescribed time constraint. The examples of such systems include control applications such as air craft control, autopilot, medical electronics, nuclear power plant control and industrial manufacturing control. So completion of every task before its deadline is very critical so for that it require an efficient scheduling algorithm. In this paper we proposed an efficient real time scheduling algorithm for preserving server budget for sporadic task set so that every job must be completed before server budget exhausted.**

**Keywords: Algorithm, RTS, Server budget.**

## I. Introduction

We propose a two phase scheduling algorithm trough utilizing the concept of multi level budget with formation of budget pattern from hyper-period to hyper-period and consume budget at multi priority level for execution of a periodic tasks while ensuring periodic one. The first phase performs offline feasibility analysis for periodic tasks and fix up static budget renovated at regular interval. Apart from static budget, slack based budget is computed forming a budget pattern from hyper period to hyper period. Over the above budget pattern formed in the first phase, the computed budget has been tuned in the second phase. The tuning is achieved trough utilizing the online slack availability arises due to execute task has taken less time, (as compared to worst case time considered at the time of feasibility analysis) accumulating the small budget fragments into larger one. The performance of the algorithms which are proposed is evaluated for both synthesized task sets used in [2] and data available for video phone application [3]. The study indicates that the proposed multi server budget receive better responsiveness with increased number of completed a periodic tasks over a wide range of variations.

Most of applications have the requirement both fixed arrival pattern as well as random pattern in nature. The fixed arrival pattern is termed as periodic task whereas term a periodic tasks used for random one. For example, mobile video phone application have stringent periodic computing requirements for number of frames received per second along with a periodic requirement generated by user such as volume control and play-list editing. These user generated request are event driven. Similarly flight control system executes pilot control commands such as control of sudden rise of temperature, speed are a periodic requirements along with routine computation requirements.

Besides these requirements patterns, time criticality classifies the system as hard and soft in nature. For hard real time system failures to met one requirement may leads to catastrophe however, degraded quality is received even missing to meet many request for the case of soft real time system.

The rest of the chapter is organized as follows: Section II, describes the system model and terminologies used. The proposed approach scheduling algorithm for server budget has been explored in Section III, results and analysis in Section IV. Finally, chapter concludes in Section V.

## II. Syatel Model and Terminology

### *A. Motivation*

With advancement in technology computing as well as storage capacity increases very rapidly and attains at very high point. However, very less improvements in battery is received than that compared with computing and storage capacity. This can be seen to conclude by author [4]. Further, over the year to year chip size reduces along with very rate of increment in computing capability. These demands accommodating more and more number of components in reduced chip area. More the number more the power consumption requirement as can be seen from summarized conclusion of [5]. The energy

consumption Pentium IV is approximately 20% of that required for the case of nuclear reactor. to improve responsiveness of a periodic tasks will require the system to operate at higher speed level as a result consume more energy and less opportunity to accumulate harvesting energy leading to miss some deadlines due to shortage of energy. Thus, responsiveness of a periodic tasks and energy issues are orthogonal.

The goal for the designer of such a system is too judiciously balance power management techniques with energy harvesting that improve the system performance in terms of reducing response time server while honoring the deadline of periodic tasks with reduced storage capacity for energy harvesting.

### *B. System Model*

Scheduling defines the sequence in which jobs of tasks are run by the processor to meet certain constraints [3, 2]. The constraints that steer the scheduling decisions depend upon the application areas and environment. For non real time applications, scheduling theory employs performance metrics such as maximizing throughput, fairness, minimization of sum of completion time, schedule length.

Fixed arrival pattern and random arrival task are known as periodic tasks and periodic tasks respectively. The periodic tasks are time driven initiated at specified time whereas occurrences of certain events are initiated at a periodic one [5]. The full guarantee can be given for periodic tasks in well advance whereas completion of a periodic one can be ensured after its arrival in online [1].

This system consists of independent mixed tasks. Every task has the values, WCET, Period, and relative deadline.

Execution of these tasks is fully preemptive and preemption overhead is considered negligible with respect to worst execution time. Here, we first summarizes the symbol used followed by definition of key terms used.

Table I: Shows Symbol Used

| Symbol | Meaning (Description) |
|---|---|
| $r_j^i$ | Release time of j th release of task $\tau_i$ |
| $D_j^i$ | Absolute deadline of j th release of task $\tau_i$ |
| $\tau_j^i$ | j th release of task $\tau_i$ |
| $ft_j^i$ | Finish time of j th release of task $\tau_i$ |
| $R_j^i$ | Response time of j th release of task $\tau_i$ |
| L | Hyper-period |

*Finish Time ($ft_j^i$):*

$$Ft_i^j = rel_i^j + e_i + \sum_{k=1}^{k=i-1} \left([t / p_k]\right) * e_k \quad \text{where } rel_i^j \le t \le p$$

## III. Proposed Algorithm for Preserving Sporadic Server Budget

Proposed algorithm has two phases for scheduling the task set. The responsibility of phase 1 a) determine feasibility of periodic tasks along with server, Ts, characterized by execution budget in term of time and renovation period of budget 1, b) decides budget 2 on the basis of slack available in the schedule of periodic task set.

The fixed amount of Ti rest budget 1 of equal to Es is allocated at each time instant equal to integral multiple of Ps. The budget released will be retain up to next due time of the allocation in case it is not consumed the server retains unconsumed budget is known as deferrable server. The server executes periodic tasks against budget 1 at the priority decided at the feasibility analysis done in offline. The server consumed budget 1, on the occurrence of either condition a) sever is executing a periodic tasks or b) system idle at the fix rate of one per unit time. The next paragraph deals with computation of budget 2. Over the above budget 1, another kind of dynamic budget is computed on the basis of slack available in the schedule of task set Ts. Here, we decides amount of dynamic budget available for specific time interval and consumed at different priority level. The amounts of budget available for consumption at different priority levels are different. The amount of budget available for consumption at priority level i is equal to the minimum of the slack available for task having priority less than or equal to Ti. The detail steps for computation of budget 2 at offline along its consumption window. The consumption window time, interval in which this computed budget is available for execution of periodic tasks along with executing priority.

Apart from consumption and replenishment rules and forming budget pattern for budget 2 from hyper-period to hyper-period, priority of server also play important role to achieving responsiveness and improve acceptance ratio for a periodic tasks. It is always desire to operate server at highest priority and have best quality in terms of responsiveness and acceptance ratio. However, on the way to increase server priority maintain the feasibility for periodic task is major problem. That is, with increased priority of server periodic task may become infeasible which were feasible without priority improvement of server.

*Algorithm I:*

Start

*For J* = 1 *to LCM* ($p_1$, $p_2$, $p_3$……$p_n$, $p_s$)/$p_n$

$B_{2,j} = L_{n,j}$ *// maximum budget 2 available in jth interval of*

*For i* = 1 *to n* - 1

*a. Compute* ($N_i = [jp_n/p_i] - ((j-1)p_n/p_i]$)

*// number of releases () of task in interval* $((j-1)p_n, Jp_n]$

*b. Compute* $L_i^j$ *using equation2*

*c. Compute* $L_{i,j} = \min(L_i^1, L_i^2, L_i^3, \ldots\ldots L_i^N$ *// Laxity of task* $\tau_i$ *in jth interval of length* $p_n$ *(Laxity*$_i$*) during* $((j-1)p_n, Jp_n]$ *// for deciding priority level for execution of budget 2*

*For* $i = 1$ *to n*

*Execution at priority level* $= i - 1$

$B_{2,j}^i = Min\ (L_{i,j}, L_{i+1,j}, L_{i+2,j} \ldots\ldots L_{n,j})$

*//amount of budget 2 executed at priority level i in j th interval of length* $p_n$

For $j = i + 1$ to $n$

$L_{i,j} = (L_{i,j} - B_{2,j}^i)$

*End for*

*End for*

*End for*

*End*

*Algorithm II:*

1**.** Initially $B_{2,j}(0) = L_{n,j}$ *// maximum budget 2 available in jth interval of* $p_n$

2. k = 1

3. $t = kPs$

4. $B_{2,j}(t) = B_{2,j}(t) + max(unused\ budget1(t) - \sum_{(k-1)P_s}^{(k)P_s} idle\ slot, 0)$

5. Update the laxity of all periodic tasks which priority is less as compare to server

6**.** Update the laxity of release which is arrive or incomplete during interval *[(k-1)Ps, kPs]*

7. $L_i^j(t) = d_i - t - reme_i(t) - \sum_{k \in \tau_H \cup \tau_s} ([d_i - t)/p_k])*(e_k)$

8. $L_{i,j}(t) = \min(L_i^j, L_i^{j+1}, L_i^{j+2}, \ldots\ldots L_i^N)$

9. $B_{2,j}^i = min\ (L_{i,j}, L_{i+1,j} L_{i+2,j}, \ldots\ldots Ln,j)$

Over the above estimation of both budget 1 and budget 2 along with priority of its execution at different priority level computed in phase 1 at offline. In phase 1 budget are computed on considering the worst case requirement in terms of both esteemed release time and worst case execution requirement. However, in online a task may not be released in estimated time, takes lesser time to complete than that considered in phase 1, and budget 1 may not be consumed up to next replenishment time. Thus on line tuning of budget 2 is discussed as below.

On line tuning of budget 2

In the view of on line variation in the requirement due to

a) Budget 1 assigned to the server has not been consumed up to next replenishment time.

b) Task releases the resource in case it takes lesser time than that considered in offline, phase 1. That is, task has been executed in worst case.

The budget 2 needs to be recomputed. Here, we are required to answer several questions, what is new dimension of budget 2? Whether budget 2 may be updated? What amount of budget is available at different priorities? What will be the new laxity for different tasks? So on.

On the way to answer these questions on line tuning is divided into two cases a) tuning due to unused budget1, b) tuning due to release took lesser time than that already booked.

For the case of unused budget; the stuffing of unused budget from one time interval to another time interval is decided on the basis of idle slots available in the previous execution window of budget 1. This can be mathematically represented as equation number 3 whereas, amount of budget available at different priorities are given in equation 4. Further, tuning of budget 2 due to task consume lesser time than that booked at the time of feasibility is given in equation 5. The priority for execution of budget 2 along with amount of budget is computed through equation.

## IV. Result and Analysis

Now we take an example and schedule according to proposed algorithm.

Considering a periodic task set $T = \{\tau_1, \tau_2\} = \{< e_i, p_i, d_i, >: < 1.5, 10, 10>, < 2.5, 15, 15>\}$ and server having attribute $\tau_s = (q_s, p_s) = (1.5, 3)$.

The seven a periodic tasks A1, A2, A3, A4, A5, A6 and A7 arrive at times 0, 3, 4.5., 6, 7.5, 8.5, and 12 with their respective execution time 0.75, 2, 1, 2, 1, 1 and 1.5. Their respective deadlines are 2, 6, 9, 10, 14, 17 and 19 units.

The slack of $\tau_2^1$ is 3 units (computed using equation 1) as a result budget 3 = 3. However, slack of $\tau_2^2$ is 4.5 units (computed using equation 1) as a result budget 2 = 3.5 units of budget 2 is available during interval (5, 30). The expiry times for budget 2 available during (0, 15] and (15, 30] are 15 and 30 respectively. On the arrival of first a periodic task, A1, at t = 0, $\tau_1^1$, $\tau_2^1$ and A1 are available in periodic and a periodic queues respectively. The priority of server is more than priority of task $\tau_1$ and $\tau_1$. However, the expiry time of budget 1 is lower than that of budget 2. Therefore, consumption of budget 1 is preferred over budget 2 and a periodic task A1 starts executing at t=0 with consumption of budget 1 while budget 2 is retained. At time t = 2, $\tau_1^2$ arrives being a higher priority it preempts a periodic A1. The periodic A1 resume at time t=3 and go ahead up to t = 3.5 and complete it with budget 1. The budget 1 is exhausted at time t=3.5. The periodic task A2 arrives at time t = 4 will be able to complete at 5 with consumption of budget 2 at priority more than $\tau_1$. However, existing deferrable server with background

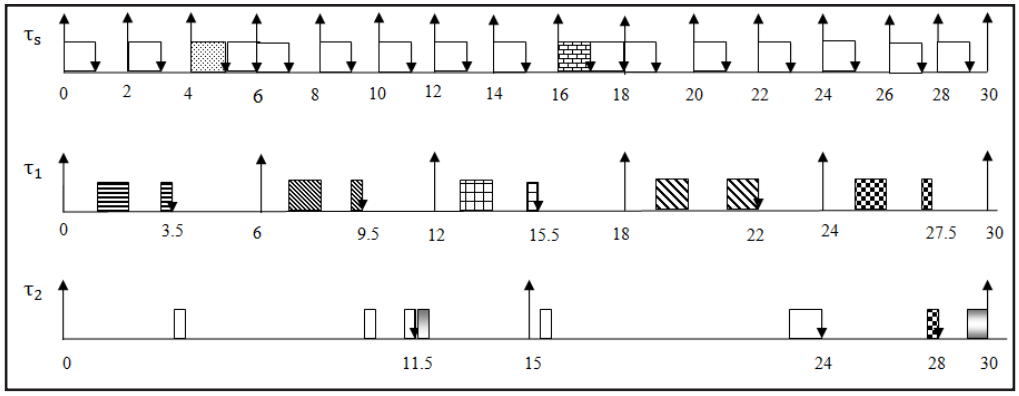


Fig. 1: Shows Schedule

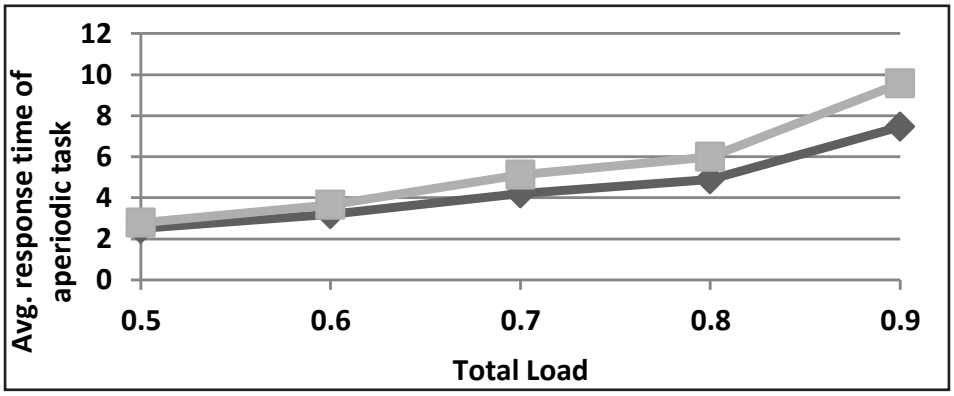


Fig. 2: ART of Aperiodic with Periodic load 40%

The effect of server utilization on the average response time of a periodic task can be seen from the Fig. 2. Fig. 2 compare the performance of proposed algorithm with existing deferrable sever when periodic load is 40% of total load and a periodic load is 20%.

## V. Conclusion

Proposed Efficient Real-time Scheduling Analytics for Preserving Sporadic Server Budget gives better result. As server budget utilization (0.11-0.112) almost 15% improvement is received due to the proposed MBBPS server provide better opportunity to a periodic task to finish it earlier due to the additional budget and consumed at multi priority level. However, at higher utilization (0.014-0.016) 5% is improvement is received because both approaches are most of the time able to finish the periodic by utilizing the budget 1.

We have proposed this sever budget algorithm has bandwidth preserving server for scheduling of mixed task set. The modified approach provides higher budget and its availability are achieved through utilizing the concept of multi budget, consumption with multi priority level with deferment.

## References

[1] L. Almeida, and P. Pedreiras, "Scheduling within temporal partitions: Response-time analysis and server design," *In Proc. of the 4th ACM Int. Conf. on Embedded Software (EMSOFT'04),* New York, NY, USA, 2004, pp. 95–103, doi:http://dx.doi.org/10.1145/1017753.1017772

[2] Ranvijay, R. S. Yadav, and S. Agrawal, "A survey on energy aware processor design techniques in real-time systems," *In Int. Conf. on Soft Computing and Intelligent Systems,* Jabalpur, Madhya Pradesh, India, pp. 27–29, December 2007.

[3] Ranvijay, R. S. Yadav, and S. Agrawal "Online speed fine tuning technique for energy minimization of mixed task set", *In INDICON 2009*, *DA-IICT*, Ahmedabad, Gujarat, India, pp. 589–592, December 2009.

[4] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," *In Proc. of 11th Real-Time Systems Symposium*, 2002.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, October 2016.